Linux
Professional
Institute

LPIC-1

Phiên bản 5.0
Tiếng Việt

101

Table of Contents

Linux Professional Institute

Chủ đề 101: Kiến trúc Hệ thống

Linux Professional Institute

101.1 Xác định và Cấu hình Cài đặt Phần cứng

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 101.1

Khối lượng

2

Các lĩnh vực kiến thức chính

- Kích hoạt và vô hiệu hóa các thiết bị ngoại vi tích hợp.
- Phân biệt các loại thiết bị lưu trữ dung lượng lớn.
- Xác định tài nguyên phần cứng cho thiết bị.
- Các công cụ và tiện ích để liệt kê các thông tin phần cứng khác nhau (ví dụ: lsusb, lspci, v.v.).
- Các công cụ và tiện ích để thao tác trên thiết bị USB.
- Hiểu về khái niệm về sysfs, udev và dbus.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

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

101.1 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	101 Kiến trúc Hệ thống
Mục tiêu:	101.1 Xác định và Định Cấu hình Cài đặt Phần Cứng
Bài:	1 trên 1

Giới thiệu

Kể từ những năm tháng đầu tiên của máy tính điện tử, các nhà sản xuất máy tính cá nhân và doanh nghiệp đã tích hợp nhiều bộ phận phần cứng khác nhau trong máy của họ. Chính vì lý do này mà chúng cần tới sự hỗ trợ của hệ điều hành. Điều này có thể khiến các nhà phát triển hệ điều hành bị choáng ngợp nếu các tiêu chuẩn cho bộ hướng dẫn và giao tiếp thiết bị không được thiết lập bởi toàn ngành. Tương tự như lớp trừu tượng được tiêu chuẩn hóa do hệ điều hành cung cấp cho một ứng dụng, các tiêu chuẩn này sẽ giúp việc viết và bảo trì một hệ điều hành không bị ràng buộc với một kiểu phần cứng cụ thể trở nên dễ dàng hơn. Tuy nhiên, sự phức tạp của các phần cứng cơ bản tích hợp đôi khi sẽ cần tới những điều chỉnh về cách các tài nguyên tiếp xúc với hệ điều hành để hệ điều hành có thể được cài đặt và hoạt động một cách chính xác.

Một số điều chỉnh có thể được thực hiện ngay cả khi chưa cài đặt hệ điều hành. Hầu hết các máy đều cung cấp tiện ích cấu hình có thể được thực thi khi máy được bật. Cho đến giữa những năm 2000, tiện ích cấu hình đã được triển khai trong BIOS (*Hệ thống Đầu vào/Đầu ra Cơ bản*), tiêu chuẩn cho phần sụn chứa các tiến trình cấu hình cơ bản có trong bo mạch chủ x86. Từ cuối thập kỷ đầu tiên của những năm 2000, các máy dựa trên kiến trúc x86 đã bắt đầu thay thế BIOS bằng một triển khai mới được gọi là UEFI (*Giao diện Phần Sụn Mở rộng Hợp nhất*) có nhiều tính năng

tinh vi hơn để nhận dạng, kiểm tra, cấu hình và nâng cấp phần sụn. Bất chấp sự thay đổi này, không quá lạ khi người ta vẫn gọi tiện ích cấu hình là BIOS vì cả hai cách triển khai đều đáp ứng cùng một mục đích cơ bản.

NOTE Thông tin chi tiết về sự giống và khác nhau giữa BIOS và UEFI sẽ được đề cập trong bài học sau.

Kích hoạt Thiết bị

Tiện ích cấu hình hệ thống được hiển thị sau khi ta nhấn một phím cụ thể lúc máy tính được bật. Phím bấm này sẽ tùy thuộc vào từng nhà sản xuất, nhưng thường sẽ là phím `Del` hoặc một trong các phím chức năng, chẳng hạn như `F2` hoặc `F12`. Tổ hợp phím để sử dụng thường sẽ được hiển thị trong màn hình bật nguồn.

Trong tiện ích thiết lập BIOS, ta có thể bật và tắt các thiết bị ngoại vi tích hợp, kích hoạt bảo vệ lỗi cơ bản và thay đổi cài đặt phần cứng như IRQ (yêu cầu ngắt) và DMA (truy cập bộ nhớ trực tiếp). Trên các máy hiện đại, việc thay đổi các cài đặt này hiếm khi cần thiết nhưng vẫn có thể cần tới một số điều chỉnh để giải quyết các vấn đề cụ thể. Chẳng hạn có những công nghệ RAM tương thích với tốc độ truyền dữ liệu nhanh hơn giá trị mặc định; vì vậy, ta nên thay đổi nó thành giá trị do nhà sản xuất chỉ định. Một số CPU sẽ cung cấp các tính năng không cần thiết cho phiên cài đặt cụ thể và có thể bị vô hiệu hóa. Các tính năng bị vô hiệu hóa sẽ giảm mức tiêu thụ điện năng và có thể tăng khả năng bảo vệ hệ thống vì các tính năng CPU chứa các lỗi đã biết cũng có thể bị vô hiệu hóa.

Nếu máy được trang bị nhiều thiết bị lưu trữ, điều quan trọng là phải xác định được thiết bị nào có trình tải khởi động chính xác và phải là mục nhập đầu tiên trong thứ tự khởi động thiết bị. Hệ điều hành có thể sẽ không tải nếu thiết bị không chính xác xuất hiện đầu tiên trong danh sách xác minh khởi động BIOS.

Kiểm tra Thiết bị trong Linux

Khi các thiết bị được xác định chính xác, hệ điều hành sẽ liên kết các thành phần phần mềm tương ứng theo yêu cầu của chúng. Khi một tính năng phần cứng không hoạt động như mong đợi, điều quan trọng là phải xác định được chính xác vấn đề đang xảy ra ở đâu. Khi một phần cứng không được hệ điều hành phát hiện, rất có thể phần đó — hoặc cổng mà nó được kết nối — đã bị lỗi. Khi phần cứng được phát hiện chính xác nhưng vẫn không thể hoạt động bình thường, có thể đã có sự cố nằm ở phía hệ điều hành. Do đó, một trong những bước đầu tiên khi xử lý các sự cố liên quan đến phần cứng là kiểm tra xem hệ điều hành có phát hiện đúng thiết bị hay không. Có hai cách cơ bản để xác định tài nguyên phần cứng trên hệ thống Linux: sử dụng các lệnh chuyên biệt hoặc đọc các tệp cụ thể bên trong các hệ thống tệp đặc biệt.

Lệnh Kiểm tra

Hai lệnh cần thiết để xác định các thiết bị được kết nối trong hệ thống Linux là:

`lspci`

Lệnh sẽ hiển thị tất cả các thiết bị hiện được kết nối với cổng PCI (*Kết nối Thiết bị Ngoại vi*). Thiết bị PCI có thể là một thành phần được gắn vào bo mạch chủ (giống như một bộ điều khiển đĩa) hoặc một thẻ mở rộng được lắp vào khe cắm PCI (ví dụ như thẻ đồ họa bên ngoài).

`lsusb`

Lệnh sẽ liệt kê các thiết bị USB (*Cổng nối Đa năng*) hiện được kết nối với máy. Mặc dù có đủ các thiết bị USB cho hầu hết mọi mục đích ta có thể nghĩ tới nhưng giao diện USB chủ yếu được sử dụng để kết nối các thiết bị đầu vào — bàn phím, thiết bị trỏ — và các phương tiện lưu trữ di động.

Đầu ra của các lệnh `lspci` và `lsusb` sẽ bao gồm một danh sách tất cả các thiết bị PCI và USB được xác định bởi hệ điều hành. Tuy nhiên, thiết bị có thể sẽ chưa hoạt động hoàn toàn vì mọi phần cứng đều yêu cầu một thành phần phần mềm để điều khiển thiết bị tương ứng. Thành phần phần mềm này được gọi là *mô-đun nhân* và nó có thể là một phần của nhân Linux chính thức hoặc được thêm riêng vào từ các nguồn khác.

Các mô-đun nhân Linux liên quan đến thiết bị phần cứng cũng được gọi là *trình điều khiển* như trong các hệ điều hành khác. Tuy nhiên, trình điều khiển của Linux không phải lúc nào cũng được nhà sản xuất thiết bị cung cấp. Trong khi một số nhà sản xuất cung cấp trình điều khiển nhị phân của riêng họ để cài đặt riêng, nhiều trình điều khiển vẫn được viết bởi các nhà phát triển độc lập. Ví dụ như trước đây, các bộ phận hoạt động trên Windows có thể sẽ không có mô-đun nhân tương ứng cho Linux. Ngày nay, các hệ điều hành dựa trên Linux đều có hỗ trợ phần cứng mạnh mẽ và hầu hết các thiết bị đều có thể hoạt động một cách dễ dàng.

Các lệnh liên quan trực tiếp đến phần cứng thường yêu cầu quyền gốc để thực thi hoặc sẽ chỉ hiển thị thông tin hạn chế khi được thực thi bởi người dùng bình thường. Vì vậy, ta sẽ cần phải đăng nhập dưới tên siêu người dùng hoặc thực thi lệnh bằng `sudo`.

Ví dụ: đầu ra sau của lệnh `lspci` sẽ hiển thị một vài thiết bị được xác định:

```
$ 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)
```

Đầu ra của các lệnh như vậy có thể dài đến hàng chục dòng. Vì vậy, ví dụ trước và ví dụ tiếp theo sẽ chỉ chứa các phần trọng tâm. Các số thập lục phân ở đầu mỗi dòng là địa chỉ duy nhất của thiết bị PCI tương ứng. Lệnh `lspci` sẽ hiển thị thêm chi tiết về một thiết bị cụ thể nếu địa chỉ của nó được cung cấp với tùy chọn `-s` kèm theo tùy chọn `-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
```

Đầu ra hiện nay đã hiển thị nhiều chi tiết hơn về thiết bị trên địa chỉ `04:02.0`. Nó là một bộ điều khiển mạng có tên nội bộ là `Ralink corp. RT2561/RT61 802.11g PCI`. Hệ thống con (`Subsystem`) được liên kết với loại và nhãn hiệu của thiết bị — `Linksys WMP54G v4.1` — và có thể sẽ hữu ích trong các mục đích chẩn đoán.

Mô-đun nhân có thể được xác định trong dòng `kernel driver in use` hiển thị mô-đun `rt61pci`. Từ tất cả các thông tin thu thập được, ta có thể giả định rằng:

1. Thiết bị đã được xác định.
2. Một mô-đun nhân phù hợp đã được tải.
3. Thiết bị sẵn sàng để sử dụng.

Một cách khác để xác minh mô-đun nhân nào đang được sử dụng cho thiết bị cụ thể có thể được cung cấp bởi tùy chọn `-k` có sẵn trong các phiên bản `lspci` gần đây:

```
$ 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
```

Đối với thiết bị đã chọn là bo mạch GPU NVIDIA, `lspci` cho ta biết rằng mô-đun đang sử dụng có tên `nvidia` tại dòng `kernel driver in use: nvidia` và tất cả các mô-đun nhân tương ứng được liệt kê trong dòng `kernel modules: nouveau , nvidia_drm, nvidia`.

Lệnh `lsusb` cũng tương tự như `lspci` nhưng nó sẽ chỉ liệt kê thông tin về 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
```

Lệnh `lsusb` sẽ hiển thị các kênh USB có sẵn và các thiết bị được kết nối với chúng. Như với `lspci`, tùy chọn `-v` sẽ hiển thị đầu ra chi tiết hơn. Ta có thể chọn một thiết bị cụ thể để kiểm tra bằng cách cung cấp ID của nó cho tùy chọn `-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
```

Với tùy chọn `-t`, lệnh `lsusb` sẽ hiển thị ánh xạ thiết bị USB hiện tại dưới dạng cây phân cấp:

```
$ 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
```

Có thể không phải tất cả các thiết bị đều có các mô-đun tương ứng được liên kết với chúng. Việc giao tiếp với một số thiết bị nhất định có thể được xử lý trực tiếp bởi ứng dụng mà không cần qua trung gian cung cấp bởi một mô-đun. Tuy nhiên, có những thông tin quan trọng trong đầu ra được cung cấp bởi `lsusb -t`. Khi một mô-đun phù hợp tồn tại, tên của nó sẽ xuất hiện ở cuối dòng dành cho thiết bị như trong `Driver=btusb`. Thiết bị `Class` sẽ xác định danh mục chung, như `Human Interface Device`, `Wireless`, `Vendor Specific Class`, `Mass Storage` và các danh mục khác. Để xác minh thiết bị nào đang sử dụng mô-đun `btusb` có trong danh sách trước, cả hai số của `Bus` và `Dev` phải được cung cấp cho tùy chọn `-s` của lệnh `lsusb`:

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

Thông thường, bất cứ lúc nào cũng có một tập hợp lớn các mô-đun nhân được tải trong một hệ thống Linux tiêu chuẩn. Cách tốt nhất để tương tác với chúng là sử dụng các lệnh được cung cấp bởi gói `kmod`. Đây là một bộ công cụ để xử lý các tác vụ phổ biến với các mô-đun nhân Linux như chèn, xóa, liệt kê, kiểm tra thuộc tính, giải quyết các phần phụ thuộc và bí danh. Ví dụ: lệnh `lsmod` sẽ hiển thị tất cả các mô-đun hiện được tải:

```
$ 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
```

Đầu ra của lệnh `lsmod` được chia thành ba cột:

`Module`

Tên mô-đun.

`Size`

Lượng RAM chiếm bởi mô-đun, tính bằng byte.

`Used by`

Các mô-đun phụ thuộc.

Một số mô-đun sẽ yêu cầu các mô-đun khác hoạt động đúng cách như trong trường hợp của các mô-đun dành cho thiết bị âm thanh:

```
$ lsmod | fgrep -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_h
da_intel,snd_seq_device
```

Cột thứ ba (`Used by`) hiển thị các mô-đun yêu cầu mô-đun trong cột đầu tiên phải hoạt động đúng cách. Nhiều mô-đun trong kiến trúc âm thanh Linux có tiền tố là `snd` sẽ phụ thuộc lẫn nhau. Khi tìm kiếm các sự cố trong quá trình chẩn đoán hệ thống, việc gỡ bỏ các mô-đun cụ thể hiện đang tải có thể sẽ rất hữu ích. Lệnh `modprobe` có thể được sử dụng để tải và dỡ bỏ các mô-đun nhân: để dỡ bỏ một mô-đun và các mô-đun liên quan của nó (miễn là chúng không được sử dụng bởi một tiến trình đang chạy), ta nên sử dụng lệnh `modprobe -r`. Ví dụ: để hủy tải mô-đun `snd-hda-intel` (mô-đun dành cho thiết bị âm thanh HDA Intel) và các mô-đun khác liên quan đến hệ thống âm thanh:

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

Ngoài việc tải và dỡ bỏ các mô-đun nhân trong khi hệ thống đang chạy, ta có thể thay đổi các tham số mô-đun khi hạt nhân đang được tải. Nó sẽ không khác lắm so với việc chuyển các tùy chọn cho các lệnh. Mỗi mô-đun sẽ chấp nhận các tham số cụ thể, nhưng hầu hết các giá trị mặc định đều được khuyến nghị và không cần thêm tham số. Tuy nhiên, trong một số trường hợp, ta sẽ cần sử dụng các tham số để thay đổi hành vi của mô-đun để chúng có thể hoạt động được như mong đợi.

Khi sử dụng tên mô-đun làm đối số duy nhất, lệnh `modinfo` sẽ hiển thị mô tả, tệp, tác giả, giấy phép, phần nhận dạng, phần phụ thuộc và các tham số khả dụng cho mô-đun đã cho. Các tham số tùy chỉnh cho một mô-đun có thể được duy trì liên tục bằng cách đưa chúng vào tệp `/etc/modprobe.conf` hoặc trong các tệp riêng lẻ có phần mở rộng `.conf` trong thư mục `/etc/modprobe.d/`. Tùy chọn `-p` sẽ làm cho lệnh `modinfo` hiển thị tất cả các tham số có sẵn và bỏ qua các thông tin khác:

```
# 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)
hdmimhz: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)
```

Đầu ra mẫu đã hiển thị tất cả các tham số có sẵn cho mô-đun `nouveau` - một mô-đun nhân do *Dự án Nouveau* cung cấp dưới dạng thay thế cho trình điều khiển độc quyền dành cho thẻ GPU NVIDIA. Ví dụ: tùy chọn `modeset` cho phép kiểm soát xem độ phân giải và độ sâu của màn hình có được đặt trong không gian hạt nhân thay vì không gian người dùng hay không. Việc thêm `options nouveau modeset=0` vào tệp `/etc/modprobe.d/nouveau.conf` sẽ vô hiệu hóa tính

năng hạt nhân modeset.

Nếu một mô-đun gây ra sự cố, tệp `/etc/modprobe.d/blacklist.conf` có thể được sử dụng để chặn việc tải mô-đun. Ví dụ: để ngăn việc tải tự động mô-đun `nouveau`, dòng `blacklist nouveau` phải được thêm vào tệp `/etc/modprobe.d/blacklist.conf`. Hành động này là bắt buộc khi mô-đun độc quyền `nvidia` được cài đặt và mô-đun mặc định `nouveau` nên được đặt sang một bên.

NOTE

Ta có thể sửa đổi tệp `/etc/modprobe.d/blacklist.conf` đã tồn tại trên hệ thống theo mặc định. Tuy nhiên, phương pháp được ưa thích là tạo một tệp cấu hình riêng là `/etc/modprobe.d/<module_name>.conf`. Tệp này sẽ chứa các cài đặt chỉ dành riêng cho mô-đun nhân nhất định.

Tệp Thông tin và Tệp Thiết bị

Các lệnh `lspci`, `lsusb` và `lsmod` đóng vai trò là giao diện phía người dùng để đọc thông tin phần cứng được hệ điều hành lưu trữ. Loại thông tin này được lưu giữ trong các tệp đặc biệt trong thư mục `/proc` và `/sys`. Các thư mục này là các điểm gắn kết với các hệ thống tệp không có trong phân vùng thiết bị mà chỉ có trong không gian RAM được hạt nhân sử dụng để lưu trữ cấu hình thời gian chạy và thông tin về các tiến trình đang chạy. Các hệ thống tệp như vậy không dành cho việc lưu trữ tệp thông thường, vì thế mà chúng được gọi là hệ thống tệp giả và chỉ tồn tại khi hệ thống đang chạy. Thư mục `/proc` sẽ chứa các tệp có thông tin liên quan đến các tiến trình đang chạy và tài nguyên phần cứng. Một số tệp quan trọng trong `/proc` để kiểm tra phần cứng là:

`/proc/cpuinfo`
: Liệt kê thông tin chi tiết về (các) CPU được tìm thấy bởi hệ điều hành.

`/proc/interrupts`
: Một danh sách các số xử lý ngắt trên mỗi thiết bị IO cho mỗi CPU.

`/proc/ioports`
: Liệt kê các khu vực cổng Đầu vào/Đầu ra đã đăng ký hiện đang được sử dụng.

`/proc/dma`
: Liệt kê các kênh DMA (truy cập bộ nhớ trực tiếp) đã đăng ký đang được sử dụng.

Các tệp bên trong thư mục `/sys` có vai trò tương tự như trong `/proc`. Tuy nhiên, thư mục `/sys` có mục đích cụ thể là lưu trữ thông tin thiết bị và dữ liệu hạt nhân liên quan đến phần cứng, trong khi `/proc` cũng chứa thông tin về các cấu trúc dữ liệu hạt nhân khác nhau bao gồm các tiến trình đang chạy và cấu hình.

Một thư mục khác liên quan trực tiếp đến các thiết bị trong hệ thống Linux tiêu chuẩn là `/dev`.

Mọi tệp bên trong `/dev` đều được liên kết với một thiết bị hệ thống, đặc biệt là các thiết bị lưu trữ. Ví dụ như khi được kết nối với kênh IDE đầu tiên của bo mạch chủ, ổ cứng IDE kế thừa sẽ được biểu thị bằng tệp `/dev/hda`. Mỗi phân vùng trong đĩa này sẽ được xác định bởi `/dev/hda1`, `/dev/hda2` cho đến phân vùng cuối cùng được tìm thấy.

Các thiết bị có thể tháo rời được xử lý bởi hệ thống con *udev*. Hệ thống này sẽ tạo ra các thiết bị tương ứng trong `/dev`. Nhân Linux sẽ nắm bắt sự kiện kiểm tra phần cứng và chuyển nó tới tiến trình udev. Tiến trình này sau đó sẽ xác định thiết bị và tự động tạo các tệp tương ứng trong `/dev` sử dụng các quy tắc đã được xác định trước.

Trong các bản phân phối Linux hiện tại, udev sẽ chịu trách nhiệm nhận dạng và cấu hình các thiết bị đã có trong quá trình khởi động máy (*phát hiện cắm nguội*) và các thiết bị được xác định trong khi hệ thống đang chạy (*phát hiện cắm nóng*). Udev dựa vào *SysFS*, tức hệ thống tệp giả, về các thông tin liên quan đến phần cứng được gắn trong `/sys`.

NOTE

Cắm nóng là thuật ngữ dùng để chỉ việc phát hiện và cấu hình thiết bị trong khi hệ thống đang chạy, chẳng hạn như khi cắm thiết bị USB. Nhân Linux đã hỗ trợ các tính năng cắm nóng kể từ phiên bản 2.6 và cho phép hầu hết các cổng hệ thống (PCI, USB, v.v.) kích hoạt các sự kiện cắm nóng khi một thiết bị được kết nối hoặc ngắt kết nối.

Khi các thiết bị mới được phát hiện, udev sẽ tìm kiếm một quy tắc phù hợp trong các quy tắc được xác định trước được lưu trữ trong thư mục `/etc/udev/rules.d/`. Các quy tắc quan trọng nhất được cung cấp bởi bản phân phối, nhưng những quy tắc mới cũng có thể được thêm vào cho các trường hợp cụ thể.

Thiết bị Lưu trữ

Trong Linux, thiết bị lưu trữ thường được gọi là thiết bị khối vì dữ liệu được đọc đến và từ các thiết bị này dưới dạng các khối dữ liệu được đệm với kích thước và vị trí khác nhau. Mỗi thiết bị khối đều được xác định bằng một tệp trong thư mục `/dev` với tên của tệp tùy thuộc vào loại thiết bị (IDE, SATA, SCSI, v.v.) và các phân vùng của nó. Ví dụ, các thiết bị CD/DVD và đĩa mềm sẽ có tên tương ứng được đặt trong `/dev`: ổ đĩa CD/DVD được kết nối với kênh IDE thứ hai sẽ được xác định là `/dev/hdc` (`/dev/hda` và `/dev/hdb` dành riêng cho thiết bị chính và phụ trên kênh IDE đầu tiên) và một ổ đĩa mềm cũ sẽ được xác định là `/dev/fd0`, `/dev/fd1`, v.v.

Kể từ nhân Linux phiên bản 2.4 trở đi, hầu hết các thiết bị lưu trữ hiện đều được xác định như thể chúng là thiết bị SCSI bất kể phần cứng của chúng thuộc loại nào. Các thiết bị khối IDE, SSD và USB sẽ có tiền tố là `sd`. Đối với các đĩa IDE, tiền tố `sd` sẽ được sử dụng, nhưng chữ cái thứ ba sẽ được chọn tùy thuộc vào việc ổ đĩa là chính hay phụ (trong kênh IDE đầu tiên, chính sẽ là `sda` và phụ sẽ là `sdb`). Các phân vùng sẽ được liệt kê bằng số. Đường dẫn `/dev/sda1`, `/dev/sda2`, v.v. được sử

dụng cho phân vùng thứ nhất và thứ hai của thiết bị khối được xác định đầu tiên; `/dev/sdb1`, `/dev/sdb2`, v.v. được sử dụng để xác định phân vùng thứ nhất và thứ hai của thiết bị khối được xác định thứ hai. Ngoại lệ đối với mẫu này là với thẻ nhớ (thẻ SD) và thiết bị NVMe (SSD được kết nối với cổng PCI Express). Đối với thẻ SD, các đường dẫn `/dev/mmcblk0p1`, `/dev/mmcblk0p2`, v.v. được sử dụng cho phân vùng thứ nhất và thứ hai của thiết bị được xác định trước; `/dev/mmcblk1p1`, `/dev/mmcblk1p2`, v.v. được sử dụng để xác định phân vùng thứ nhất và thứ hai của thiết bị được xác định thứ hai. Các thiết bị NVMe sẽ nhận tiền tố `nvme` như trong `/dev/nvme0n1p1` và `/dev/nvme0n1p2`.

Bài tập Hướng dẫn

1. Giả sử một hệ điều hành không thể khởi động sau khi thêm đĩa SATA thứ hai vào hệ thống, biết rằng tất cả các bộ phận đều hoạt động bình thường. Nguyên nhân có thể là gì?

2. Giả sử bạn muốn đảm bảo rằng thẻ video bên ngoài được kết nối với cổng PCI của máy tính để bàn mới mua của bạn thực sự là thẻ được nhà sản xuất quảng cáo, nhưng việc mở vỏ máy tính sẽ làm mất hiệu lực bảo hành. Lệnh nào có thể được sử dụng để liệt kê các chi tiết của thẻ màn hình khi chúng được hệ điều hành phát hiện?

3. Dòng sau đây là một phần của đầu ra được tạo bởi lệnh `lspci`:

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

   Bạn nên thực hiện lệnh nào để xác định mô-đun nhân đang được sử dụng cho thiết bị cụ thể này?

4. Quản trị viên hệ thống muốn thử các tham số khác nhau cho mô-đun nhân `bluetooth` mà không cần khởi động lại hệ thống. Tuy nhiên, bất kỳ cách nào để hủy tải mô-đun bằng `modprobe -r bluetooth` đều dẫn đến lỗi sau:

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

   Nguyên nhân có thể là gì?

Bài tập Mở rộng

1. Không có gì lạ khi tìm thấy các máy cũ trong môi trường sản xuất, chẳng hạn như khi một số thiết bị sử dụng kết nối lỗi thời để giao tiếp với máy tính điều khiển khiến ta cần phải đặc biệt chú ý đến một số đặc thù của những máy cũ này. Ví dụ, một số máy chủ x86 có phần sụn BIOS cũ hơn sẽ không khởi động nếu không phát hiện thấy bàn phím. Làm thế nào để tránh được vấn đề này?

2. Các hệ điều hành được xây dựng xung quanh nhân Linux cũng có sẵn cho nhiều loại kiến trúc máy tính khác ngoài x86, chẳng hạn như trong các máy tính bảng đơn dựa trên kiến trúc ARM. Một người dùng cẩn thận sẽ nhận thấy sự vắng mặt của lệnh `lspci` trên các máy như Raspberry Pi. Sự khác biệt nào với các máy x86 có thể giải thích cho sự vắng mặt này?

3. Nhiều bộ định tuyến mạng có cổng USB cho phép kết nối với thiết bị bên ngoài, chẳng hạn như ổ cứng USB. Vì hầu hết trong số này đang sử dụng hệ điều hành dựa trên Linux, ổ cứng USB bên ngoài sẽ được đặt tên như thế nào trong thư mục `/dev/` (giả sử không có thiết bị khối thông thường nào khác có trong bộ định tuyến)?

4. Vào năm 2018, lỗ hổng phần cứng có tên *Meltdown* đã được phát hiện. Nó ảnh hưởng đến hầu hết tất cả các bộ xử lý của nhiều kiến trúc. Các phiên bản gần đây của nhân Linux có thể thông báo liệu hệ thống hiện tại có dễ bị tấn công hay không. Làm thế nào để có được thông tin này?

Tóm tắt

Bài học này đã trình bày các khái niệm chung về cách nhân Linux xử lý tài nguyên phần cứng, chủ yếu trong kiến trúc x86. Bài học đã đi qua các chủ đề sau:

- Cách cài đặt được xác định trong tiện ích cấu hình BIOS hoặc UEFI có thể ảnh hưởng đến cách hệ điều hành tương tác với phần cứng.
- Cách sử dụng các công cụ do hệ thống Linux tiêu chuẩn cung cấp để lấy thông tin về phần cứng.
- Cách xác định các thiết bị lưu trữ vĩnh viễn và di động trong hệ thống tệp.

Các lệnh và tiến trình đã được nhắc đến là:

- Các lệnh kiểm tra phần cứng được phát hiện: `lspci` và `lsusb`.
- Các lệnh quản lý mô-đun nhân: `lsmod` và `modprobe`.
- Các tệp đặc biệt liên quan đến phần cứng (hoặc là các tệp được tìm thấy trong thư mục `/dev/` hoặc trong các hệ thống tệp giả trong `/proc/` và `/sys/`).

Đáp án Bài tập Hướng dẫn

1. Giả sử một hệ điều hành không thể khởi động sau khi thêm đĩa SATA thứ hai vào hệ thống, biết rằng tất cả các bộ phận đều hoạt động bình thường. Nguyên nhân có thể là gì?

   Thứ tự thiết bị khởi động phải được cấu hình trong tiện ích thiết lập BIOS. Nếu không, BIOS có thể sẽ không chạy được trình tải khởi động.

2. Giả sử bạn muốn đảm bảo rằng thẻ video bên ngoài được kết nối với cổng PCI của máy tính để bàn mới mua của bạn thực sự là thẻ được nhà sản xuất quảng cáo, nhưng việc mở vỏ máy tính sẽ làm mất hiệu lực bảo hành. Lệnh nào có thể được sử dụng để liệt kê các chi tiết của thẻ màn hình khi chúng được hệ điều hành phát hiện?

   Lệnh `lspci` sẽ liệt kê thông tin chi tiết về tất cả các thiết bị hiện được kết nối với cổng PCI.

3. Dòng sau đây là một phần của đầu ra được tạo bởi lệnh `lspci`:

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

   Bạn nên thực hiện lệnh nào để xác định mô-đun nhân đang được sử dụng cho thiết bị cụ thể này?

   Lệnh `lspci -s 03:00.0 -v` hoặc `lspci -s 03:00.0 -k`

4. Quản trị viên hệ thống muốn thử các tham số khác nhau cho mô-đun nhân `bluetooth` mà không cần khởi động lại hệ thống. Tuy nhiên, bất kỳ cách nào để hủy tải mô-đun bằng `modprobe -r bluetooth` đều dẫn đến lỗi sau:

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

   Nguyên nhân có thể là gì?

   Mô-đun `bluetooth` đang được sử dụng bởi một tiến trình đang chạy.

Đáp án Bài tập Mở rộng

1. Không có gì lạ khi tìm thấy các máy cũ trong môi trường sản xuất, chẳng hạn như khi một số thiết bị sử dụng kết nối lỗi thời để giao tiếp với máy tính điều khiển khiến ta cần phải đặc biệt chú ý đến một số đặc thù của những máy cũ này. Ví dụ, một số máy chủ x86 có phần sụn BIOS cũ hơn sẽ không khởi động nếu không phát hiện thấy bàn phím. Làm thế nào để tránh được vấn đề này?

   Tiện ích cấu hình BIOS có một tùy chọn để tắt khóa máy tính khi không tìm thấy bàn phím.

2. Các hệ điều hành được xây dựng xung quanh nhân Linux cũng có sẵn cho nhiều loại kiến trúc máy tính khác ngoài x86, chẳng hạn như trong các máy tính bảng đơn dựa trên kiến trúc ARM. Một người dùng cẩn thận sẽ nhận thấy sự vắng mặt của lệnh `lspci` trên các máy như Raspberry Pi. Sự khác biệt nào với các máy x86 có thể giải thích cho sự vắng mặt này?

   Không giống như hầu hết các máy x86, một máy tính dựa trên ARM như Raspberry Pi không có cổng PCI, vì thế nên lệnh `lspci` sẽ là vô dụng.

3. Nhiều bộ định tuyến mạng có cổng USB cho phép kết nối với thiết bị bên ngoài, chẳng hạn như ổ cứng USB. Vì hầu hết trong số này đang sử dụng hệ điều hành dựa trên Linux, ổ cứng USB bên ngoài sẽ được đặt tên như thế nào trong thư mục `/dev/` (giả sử không có thiết bị khối thông thường nào khác có trong bộ định tuyến)?

   Các nhân Linux hiện đại xác định ổ cứng USB là thiết bị SATA. Do đó, tệp tương ứng sẽ là `/dev/sda` vì không có thiết bị khối thông thường nào khác tồn tại trong hệ thống.

4. Vào năm 2018, lỗ hổng phần cứng có tên *Meltdown* đã được phát hiện. Nó ảnh hưởng đến hầu hết tất cả các bộ xử lý của nhiều kiến trúc. Các phiên bản gần đây của nhân Linux có thể thông báo liệu hệ thống hiện tại có dễ bị tấn công hay không. Làm thế nào để có được thông tin này?

   Tệp `/proc/cpuinfo` có một dòng hiển thị các lỗi đã biết đối với CPU tương ứng như `bugs: cpu_meltdown`.

101.2 Khởi động Hệ thống

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 101.2

Khối lượng

3

Các lĩnh vực kiến thức chính

- Cung cấp các lệnh chung cho trình tải khởi động và các tùy chọn cho hạt nhân khi khởi động.
- Thể hiện kiến thức về trình tự khởi động từ BIOS/UEFI đến khi hoàn tất quá trình khởi động.
- Hiểu về SysVinit và systemd.
- Hiểu về về Upstart.
- Kiểm tra các sự kiện khởi động trong tệp nhật ký.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

- `dmesg`
- `journalctl`
- BIOS
- UEFI
- bootloader
- kernel
- `initramfs`
- `init`
- SysVinit

- systemd

101.2 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	101 Kiến trúc Hệ thống
Mục tiêu:	101.2 Khởi động Hệ thống
Bài:	1 trên 1

Giới thiệu

Để quản lý một máy tính, thành phần chính của hệ điều hành — tức hạt nhân — phải được tải bởi một chương trình có tên là *trình tải khởi động*. Bản thân chương trình này sẽ được tải bởi phần sụn được cài đặt sẵn như BIOS hoặc UEFI. Trình tải khởi động có thể được tùy chỉnh để chuyển các tham số cho hạt nhân (chẳng hạn như phân vùng nào chứa hệ thống tệp gốc hoặc hệ điều hành sẽ thực thi ở chế độ nào). Sau khi được tải, hạt nhân sẽ tiếp tục quá trình khởi động để xác định và cấu hình phần cứng. Cuối cùng, hạt nhân sẽ gọi lên tiện ích chịu trách nhiệm khởi động và quản lý các dịch vụ của hệ thống.

NOTE Trên một số bản phân phối Linux, các lệnh được thực hiện trong bài học này có thể sẽ yêu cầu quyền gốc.

BIOS hay UEFI

Các quy trình được thực hiện bởi các máy x86 để chạy trình tải khởi động đều sẽ khác nhau dù chúng sử dụng BIOS hay UEFI. BIOS (viết tắt của *Basic Input/Output System* - Hệ thống Đầu vào/Đầu ra Cơ bản) là một chương trình được lưu trữ trong một chip bộ nhớ cố định được gắn vào bo mạch chủ và sẽ được thực thi mỗi khi ta bật máy tính. Loại chương trình này được gọi là *phần sụn* và vị trí lưu trữ của nó sẽ tách biệt với các thiết bị lưu trữ khác mà hệ thống có thể có. BIOS giả định rằng 440 byte đầu tiên trong thiết bị lưu trữ đầu tiên — theo thứ tự được xác định trong tiện ích cấu hình BIOS — sẽ là giai đoạn đầu tiên của trình tải khởi động (bootloader, hay còn được gọi là *bootstrap*). 512 byte đầu tiên của thiết bị lưu trữ được gọi là MBR (*Bản ghi Khởi động Chính*) của thiết bị lưu trữ sử dụng lược đồ phân vùng DOS tiêu chuẩn và còn chứa cả bảng phân vùng bên cạnh giai đoạn đầu tiên của trình tải khởi động. Nếu MBR không chứa dữ liệu chính xác, hệ thống sẽ không thể khởi động trừ khi ta sử dụng một phương pháp thay thế.

Nói chung, các bước trước khi vận hành để khởi động hệ thống được trang bị BIOS là:

1. Tiến trình POST (power-on self-test - *tự kiểm tra khi bật nguồn*) sẽ được thực thi để xác định các lỗi phần cứng đơn giản ngay khi bật nguồn máy.
2. BIOS sẽ kích hoạt các thành phần cơ bản để tải hệ thống như đầu ra video, bàn phím và phương tiện lưu trữ.
3. BIOS sẽ tải giai đoạn đầu tiên của trình tải khởi động từ MBR (440 byte đầu tiên của thiết bị đầu tiên như được xác định trong tiện ích cấu hình BIOS).
4. Giai đoạn đầu tiên của trình tải khởi động sẽ gọi giai đoạn thứ hai của trình tải khởi động chịu trách nhiệm trình bày các tùy chọn khởi động và tải hạt nhân.

UEFI (viết tắt của *Unified Extensible Firmware Interface* - Giao diện Phần sụn Mở rộng Hợp nhất) khác với BIOS ở một số điểm trọng yếu. Giống như BIOS, UEFI cũng là một phần sụn nhưng nó có thể xác định các phân vùng và đọc nhiều hệ thống tệp được tìm thấy trong đó. UEFI không dựa vào MBR mà chỉ tính đến các cài đặt được lưu trữ trong bộ nhớ cố định (*NVRAM*) được gắn vào bo mạch chủ. Các định nghĩa này cho ta biết về vị trí của các chương trình tương thích với UEFI (được gọi là *ứng dụng EFI*) sẽ được thực thi tự động hoặc được gọi từ menu khởi động. Các ứng dụng EFI có thể là trình tải khởi động, bộ chọn hệ điều hành, công cụ chẩn đoán và sửa chữa hệ thống, v.v. Chúng phải nằm trong phân vùng thiết bị lưu trữ thông thường và trong một hệ thống tệp tương thích. Các hệ thống tệp tương thích tiêu chuẩn là FAT12, FAT16 và FAT32 cho các thiết bị khối và ISO-9660 cho các phương tiện quang học. Cách tiếp cận này cho phép ta triển khai nhiều công cụ phức tạp hơn so với những công cụ mà ta có thể có với BIOS.

Phân vùng chứa các ứng dụng EFI được gọi là *Phân vùng hệ thống EFI* hoặc ESP (EFI System Partition). Phân vùng này không được chia sẻ với các hệ thống tệp hệ thống khác (chẳng hạn như

hệ thống tệp gốc hoặc hệ thống tệp dữ liệu người dùng). Thư mục EFI trong phân vùng ESP sẽ chứa các ứng dụng được trỏ đến bởi các mục được lưu trong NVRAM.

Nói chung, các bước trước khởi động hệ điều hành trên hệ thống có UEFI là:

1. Tiến trình POST được thực thi để xác định các lỗi phần cứng đơn giản ngay khi bật nguồn máy.
2. UEFI sẽ kích hoạt các thành phần cơ bản để tải hệ thống như đầu ra video, bàn phím và phương tiện lưu trữ.
3. Phần sụn của UEFI sẽ đọc các định nghĩa được lưu trữ trong NVRAM để thực thi ứng dụng EFI được xác định trước và được lưu trữ trong hệ thống tệp của phân vùng ESP. Thông thường, ứng dụng EFI được xác định trước chính là trình tải khởi động.
4. Nếu ứng dụng EFI được xác định trước là trình tải khởi động, nó sẽ tải hạt nhân để khởi động hệ điều hành.

Tiêu chuẩn UEFI cũng hỗ trợ một tính năng gọi là *Khởi động An toàn*. Tính năng này chỉ cho phép thực thi các ứng dụng EFI đã ký, tức là các ứng dụng EFI được nhà sản xuất phần cứng ủy quyền. Tính năng này sẽ tăng khả năng bảo vệ chống lại các phần mềm độc hại, nhưng nó cũng có thể gây khó khăn cho việc cài đặt các hệ điều hành không được bảo hành của nhà sản xuất.

Trình Tải Khởi động

Trình tải khởi động phổ biến nhất của Linux trong kiến trúc x86 là GRUB (*Grand Unified Bootloader* - Bộ Tải Khởi động Hợp nhất Lớn). Ngay khi được BIOS hoặc UEFI gọi, GRUB sẽ hiển thị một danh sách các hệ điều hành có sẵn để khởi động. Đôi khi danh sách sẽ không tự động xuất hiện nhưng nó có thể được gọi bằng cách nhấn phím `Shift` trong khi GRUB đang được BIOS gọi. Trong các hệ thống UEFI, ta nên sử dụng phím `Esc` để thay thế.

Từ menu GRUB, ta có thể chọn một trong các hạt nhân đã cài đặt sẽ được tải và chuyển các tham số mới cho hạt nhân đó. Hầu hết các tham số hạt nhân đều sẽ tuân theo mẫu `option=value`. Một số tham số hạt nhân hữu ích nhất là:

`acpi`

Bật/tắt hỗ trợ ACPI. `acpi=off` sẽ tắt hỗ trợ cho ACPI.

`init`

Đặt một bộ khởi tạo hệ thống thay thế. Ví dụ: `init=/bin/bash` sẽ đặt vỏ Bash làm trình khởi tạo. Điều này có nghĩa là phiên vỏ sẽ bắt đầu ngay sau quá trình khởi động hạt nhân.

`systemd.unit`

Đặt kích hoạt mục tiêu *systemd* (ví dụ như `systemd.unit=graphical.target`). Systemd cũng

chấp nhận các mức chạy số như được xác định cho *SysV*. Ví dụ: để kích hoạt mức chạy 1, ta chỉ cần thêm số `1` hoặc chữ cái `S` (viết tắt của "single") làm tham số hạt nhân.

`mem`

Đặt dung lượng RAM khả dụng cho hệ thống. Tham số này sẽ rất hữu ích cho các máy ảo trong việc giới hạn dung lượng RAM khả dụng cho mỗi máy khách. Việc sử dụng `mem=512M` sẽ giới hạn dung lượng RAM khả dụng ở mức 512 megabyte cho một hệ thống khách cụ thể.

`maxcpus`

Giới hạn số lượng bộ xử lý (hoặc lõi bộ xử lý) hiển thị cho hệ thống trong các máy đa bộ xử lý đối xứng. Nó cũng hữu ích cho các máy ảo. Giá trị `0` sẽ tắt hỗ trợ cho các máy đa bộ xử lý và có tác dụng tương tự như tham số hạt nhân `nosmp`. Tham số `maxcpus=2` sẽ giới hạn số lượng bộ xử lý có sẵn cho hệ điều hành xuống còn hai.

`quiet`

Ẩn hầu hết các thông báo khởi động.

`vga`

Chọn một chế độ video. Tham số `vga=ask` sẽ hiển thị danh sách các chế độ khả dụng để chọn.

`root`

Đặt phân vùng gốc khác với phân vùng được định cấu hình sẵn trong trình tải khởi động. Ví dụ: `root=/dev/sda3`.

`rootflags`

Tùy chọn gắn kết cho hệ thống tệp gốc.

`ro`

Đặt lần gắn ban đầu của hệ thống tệp gốc ở chế độ chỉ đọc.

`rw`

Cho phép ghi vào hệ thống tệp gốc trong quá trình gắn kết ban đầu.

Việc thay đổi các tham số hạt nhân thường không bắt buộc, nhưng nó có thể sẽ hữu ích trong việc phát hiện và giải quyết các sự cố liên quan đến hệ điều hành. Các tham số hạt nhân phải được thêm vào tệp `/etc/default/grub` trong dòng `GRUB_CMDLINE_LINUX` để khiến chúng được ổn định trong các lần khởi động lại. Một tệp cấu hình mới cho trình tải khởi động phải được tạo mỗi khi `/etc/default/grub` thay đổi. Việc này được thực hiện bằng lệnh `grub-mkconfig -o /boot/grub/grub.cfg`. Một khi hệ điều hành đã chạy, các tham số hạt nhân được sử dụng để tải phiên hiện tại sẽ có sẵn để đọc trong tệp `/proc/cmdline`.

NOTE Cấu hình GRUB sẽ được thảo luận thêm trong bài học sau.

Khởi tạo Hệ thống

Ngoài hạt nhân, hệ điều hành cũng phụ thuộc vào các thành phần khác trong việc cung cấp các tính năng được mong đợi. Nhiều thành phần trong số này sẽ được tải trong quá trình khởi tạo hệ thống, từ các tệp lệnh vỏ đơn giản đến các chương trình dịch vụ phức tạp hơn. Các tệp lệnh thường được sử dụng để thực hiện các tác vụ ngắn sẽ chạy và chấm dứt trong quá trình khởi tạo hệ thống. Các dịch vụ này còn được gọi là *trình nền* (daemon) và có thể hoạt động mọi lúc vì chúng có thể chịu trách nhiệm về các khía cạnh nội tại của hệ điều hành.

Các kết hợp giữa các tệp lệnh khởi động và các trình nền với nhiều đặc điểm khác biệt vào một bản phân phối Linux là vô cùng đa dạng. Đây là một thực tế trong lịch sử đã cản trở sự phát triển của một giải pháp duy nhất đáp ứng được kỳ vọng của những nhà bảo trì và người dùng của tất cả các bản phân phối Linux. Tuy nhiên, bất kỳ công cụ nào mà những nhà bảo trì phân phối đã chọn để thực hiện chức năng này ít nhất sẽ có thể bắt đầu, dừng và khởi động lại các dịch vụ hệ thống. Ví dụ, những tác vụ này thường được thực hiện bởi chính hệ thống sau khi cập nhật phần mềm, nhưng quản trị viên hệ thống hầu như luôn cần phải khởi động lại dịch vụ theo cách thủ công sau khi thực hiện các sửa đổi đối với tệp cấu hình của nó.

Nó cũng sẽ thuận tiện cho người quản trị hệ thống để có thể kích hoạt một bộ trình nền cụ thể, tùy thuộc vào hoàn cảnh (ví dụ như việc ta có thể chỉ chạy một bộ dịch vụ tối thiểu để thực hiện các tác vụ bảo trì hệ thống).

NOTE Nói một cách chính xác, hệ điều hành chỉ là một hạt nhân cùng với các thành phần của nó với mục đích điều khiển phần cứng và quản lý tất cả các tiến trình. Tuy nhiên, người ta thường sử dụng thuật ngữ “hệ điều hành” một cách lỏng lẻo hơn để chỉ toàn bộ một nhóm các chương trình riêng biệt tạo nên môi trường phần mềm nơi người dùng có thể thực hiện các tác vụ tính toán cơ bản.

Quá trình khởi tạo hệ điều hành sẽ bắt đầu khi trình tải khởi động tải hạt nhân vào RAM. Sau đó, hạt nhân sẽ phụ trách CPU và bắt đầu phát hiện và thiết lập các khía cạnh cơ bản của hệ điều hành như cấu hình phần cứng cơ bản và địa chỉ bộ nhớ.

Sau đó, hạt nhân sẽ mở *initramfs* (*initial RAM filesystem* - hệ thống tệp RAM ban đầu). initramfs là kho lưu trữ chứa một hệ thống tệp được sử dụng làm hệ thống tệp gốc tạm thời trong quá trình khởi động. Mục đích chính của một tệp initramfs là cung cấp các mô-đun cần thiết để hạt nhân có thể truy cập hệ thống tệp gốc “thực” của hệ điều hành.

Ngay khi hệ thống tệp gốc khả dụng, hạt nhân sẽ gắn kết tất cả các hệ thống tệp được định cấu hình trong `/etc/fstab` và sau đó sẽ thực thi chương trình đầu tiên - một tiện ích có tên là `init`.

Chương trình `init` sẽ chịu trách nhiệm chạy tất cả các tệp lệnh khởi tạo và các trình nền hệ thống. Ngoài bộ khởi tạo truyền thống, ta còn có các triển khai riêng biệt của các bộ khởi tạo hệ thống như *systemd* và *Upstart*. Khi chương trình init được tải, initramfs sẽ bị xóa khỏi RAM.

SysV standard

Trình quản lý dịch vụ dựa trên tiêu chuẩn SysVinit sẽ kiểm soát trình nền và tài nguyên nào sẽ khả dụng bằng cách sử dụng khái niệm *mức chạy*. Các mức chạy sẽ được đánh số từ 0 đến 6 và được thiết kế bởi các nhà bảo trì phân phối để đáp ứng các mục đích cụ thể. Các định nghĩa mức chạy duy nhất được dùng chung bởi tất cả các bản phân phối là các mức chạy 0, 1 và 6.

systemd

systemd là một trình quản lý dịch vụ và hệ thống hiện đại với một lớp tương thích cho các lệnh và mức chạy SysV. systemd có cấu trúc đồng thời sử dụng ổ cắm và D-Bus để kích hoạt dịch vụ, thực thi trình nền theo yêu cầu, giám sát tiến trình với *cgroups*, hỗ trợ chụp nhanh hình ảnh tức thời, khôi phục phiên hệ thống, kiểm soát điểm gắn kết và kiểm soát dịch vụ dựa trên phần phụ thuộc. Trong những năm gần đây, hầu hết các bản phân phối Linux lớn đều đã dần dần sử dụng systemd làm trình quản lý hệ thống mặc định.

Upstart

Giống như systemd, Upstart là một giải pháp thay thế cho init. Trọng tâm của Upstart là tăng tốc quá trình khởi động bằng cách song song hóa quá trình tải các dịch vụ hệ thống. Upstart đã được sử dụng bởi các bản phân phối dựa trên Ubuntu trong các bản phát hành trước đây, nhưng ngày nay cũng đã nhường chỗ cho systemd.

Kiểm tra Quá trình Khởi tạo

Trong quá trình khởi động, lỗi luôn có thể xảy ra nhưng chúng có thể không nghiêm trọng đến mức khiến cho hệ điều hành đình chỉ hoàn toàn. Mặc dù vậy, những lỗi này có thể sẽ ảnh hưởng đến những hoạt động thông thường của hệ thống. Tất cả các lỗi đều sẽ dẫn đến các thông báo có thể được sử dụng cho các cuộc điều tra trong tương lai vì chúng có chứa các thông tin giá trị về thời gian và cách thức xảy ra lỗi. Ngay cả khi không có thông báo lỗi nào được tạo, thông tin được thu thập trong quá trình khởi động có thể vẫn sẽ hữu ích cho mục đích điều chỉnh và cấu hình.

Không gian bộ nhớ nơi hạt nhân lưu trữ các thông báo của nó (bao gồm cả thông báo khởi động) được gọi là *bộ đệm vòng hạt nhân*. Các thông báo sẽ được giữ trong bộ đệm vòng hạt nhân ngay cả khi chúng không được hiển thị trong quá trình khởi tạo, chẳng hạn như khi một hoạt ảnh được hiển thị thay thế. Tuy nhiên, bộ đệm vòng hạt nhân sẽ mất tất cả các thông báo khi hệ thống bị tắt hoặc khi ta thực hiện lệnh `dmesg --clear`. Nếu không có tùy chọn, lệnh `dmesg` sẽ hiển thị các thông báo hiện tại trong bộ đệm vòng hạt nhân:

```
$ 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
```

Đầu ra của `dmesg` có thể dài đến hàng trăm dòng. Vì vậy, danh sách trên sẽ chỉ chứa đoạn trích hiển thị quá trình hạt nhân gọi trình quản lý dịch vụ systemd. Các giá trị ở đầu dòng là số giây tương ứng với thời điểm bắt đầu tải hạt nhân.

Trong các hệ thống dựa trên systemd, lệnh `journalctl` sẽ hiển thị thông báo khởi tạo với các tùy chọn `-b`, `--boot`, `-k` hoặc `--dmesg`. Lệnh `journalctl --list-boots` sẽ hiển thị danh sách các số khởi động liên quan đến lần khởi động hiện tại, hàm băm nhận dạng của chúng và dấu thời gian của thông báo tương ứng đầu tiên và cuối cùng:

```
$ 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 55c0d9439bfb4e85a20a62776d0dbb4d 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
```

Các bản ghi khởi tạo trước đó cũng được lưu giữ trong các hệ thống dựa trên systemd. Vì vậy, các thông báo từ các phiên hệ điều hành trước đó vẫn có thể được kiểm tra. Nếu các tùy chọn `-b 0` hoặc `--boot=0` được cung cấp thì các thông báo cho lần khởi động hiện tại sẽ được hiển thị. Tùy

chọn `-b -1` hoặc `--boot=-1` sẽ hiển thị các thông báo từ lần khởi tạo trước. Các tùy chọn `-b -2` hoặc `--boot=-2` sẽ hiển thị các thông báo từ lần khởi tạo trước đó, v.v. Đoạn trích sau đây sẽ cho thấy quá trình hạt nhân gọi trình quản lý dịch vụ systemd cho quá trình khởi tạo cuối cùng:

```
$ 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...
```

Quá trình khởi tạo và các thông báo khác do hệ điều hành đưa ra sẽ được lưu trữ trong các tệp bên trong thư mục `/var/log/`. Nếu một lỗi nghiêm trọng xảy ra và hệ điều hành không thể tiếp tục quá trình khởi tạo sau khi tải hạt nhân và initramfs, một phương tiện khởi động thay thế có thể được sử dụng để khởi động hệ thống và truy cập hệ thống tệp tương ứng. Sau đó, các tệp trong `/var/log/` sẽ có thể được tìm kiếm để xác định các nguyên nhân có thể gây ra sự gián đoạn quá trình khởi động. Các tùy chọn `-D` hoặc `--directory` của lệnh `journalctl` có thể được sử dụng để đọc các thông điệp tường trình trong các thư mục khác với `/var/log/journal/` - vị trí mặc định cho các thông báo nhật ký của systemd. Vì các thông báo nhật ký của systemd không được lưu trữ

ở dạng văn bản thô nên ta cần có lệnh `journalctl` để có thể đọc chúng.

Bài tập Hướng dẫn

1. Trên một máy có phần sụn BIOS, tệp nhị phân bootstrap sẽ nằm ở đâu?

2. Phần sụn UEFI hỗ trợ các tính năng mở rộng được cung cấp bởi các chương trình bên ngoài được gọi là ứng dụng EFI. Tuy nhiên, những ứng dụng này có vị trí đặc biệt của riêng chúng. Các ứng dụng EFI sẽ được đặt ở đâu trong hệ thống?

3. Trình tải khởi động cho phép truyền các tham số hạt nhân tùy chỉnh trước khi tải nó. Giả sử hệ thống không thể khởi động do vị trí hệ thống tệp gốc bị sai thông tin, làm cách nào để hệ thống tệp gốc chính xác (nằm ở `/dev/sda3`) có thể được cung cấp làm tham số cho hạt nhân?

4. Quá trình khởi động của máy Linux kết thúc với thông báo sau:

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

   Nguyên nhân của vấn đề này là gì?

Bài tập Mở rộng

1. Trình tải khởi động sẽ hiển thị danh sách các hệ điều hành để chọn khi có nhiều hơn một hệ điều hành được cài đặt trên máy. Tuy nhiên, một hệ điều hành mới được cài đặt có thể sẽ ghi đè lên MBR của ổ đĩa cứng, xóa giai đoạn đầu tiên của trình tải khởi động và khiến hệ điều hành khác không thể truy cập được. Tại sao điều này lại không xảy ra trên máy có phần sụn UEFI?

2. Một hậu quả phổ biến của việc cài đặt hạt nhân tùy chỉnh mà không cung cấp hình ảnh initramfs thích hợp là gì?

3. Nhật ký khởi tạo có thể dài đến hàng trăm dòng. Vì vậy, đầu ra của lệnh `dmesg` thường được chuyển thành lệnh máy nhắn tin — như lệnh `less` — để thuận tiện cho việc đọc. Tùy chọn `dmesg` nào sẽ tự động phân trang đầu ra của nó và loại bỏ nhu cầu sử dụng lệnh máy nhắn tin một cách rõ ràng?

4. Ổ đĩa cứng chứa toàn bộ hệ thống tệp của máy ngoại tuyến đã bị xóa và gắn vào máy đang hoạt động dưới dạng ổ đĩa phụ. Giả sử điểm gắn kết của nó là `/mnt/hd` thì `journalctl` sẽ được sử dụng như thế nào để kiểm tra nội dung của các tệp nhật ký nằm ở `/mnt/hd/var/log/journal/`?

Tóm tắt

Bài học này đã nói về trình tự khởi động trong một hệ thống Linux tiêu chuẩn. Kiến thức chính xác về cách hoạt động của tiến trình khởi động của hệ thống Linux sẽ giúp ngăn chặn các lỗi có thể xảy ra khiến hệ thống không thể truy cập được. Bài học đã đi qua các lĩnh vực chủ đề sau:

- Phương pháp khởi động BIOS và UEFI khác nhau như thế nào.
- Các giai đoạn khởi tạo hệ thống điển hình.
- Khôi phục thông báo khởi động.

Các lệnh và tiến trình đã được nhắc đến là:

- Các tham số hạt nhân phổ biến.
- Các lệnh để đọc thông báo khởi động: `dmesg` và `journalctl`.

Đáp án Bài tập Hướng dẫn

1. Trên một máy có phần sụn BIOS, tệp nhị phân bootstrap sẽ nằm ở đâu?

   Trong MBR của thiết bị lưu trữ đầu tiên như đã được định nghĩa trong tiện ích cấu hình BIOS.

2. Phần sụn UEFI hỗ trợ các tính năng mở rộng được cung cấp bởi các chương trình bên ngoài được gọi là ứng dụng EFI. Tuy nhiên, những ứng dụng này có vị trí đặc biệt của riêng chúng. Các ứng dụng EFI sẽ được đặt ở đâu trong hệ thống?

   Các ứng dụng EFI được lưu trữ trong Phân vùng hệ thống EFI (ESP) nằm ở bất kỳ khối lưu trữ khả dụng nào có hệ thống tệp tương thích (thường là hệ thống tệp FAT32).

3. Trình tải khởi động cho phép truyền các tham số hạt nhân tùy chỉnh trước khi tải nó. Giả sử hệ thống không thể khởi động do vị trí hệ thống tệp gốc bị sai thông tin, làm cách nào để hệ thống tệp gốc chính xác (nằm ở `/dev/sda3`) có thể được cung cấp làm tham số cho hạt nhân?

   Nên sử dụng tham số `root` như trong `root=/dev/sda3`.

4. Quá trình khởi động của máy Linux kết thúc với thông báo sau:

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

   Nguyên nhân của vấn đề này là gì?

   Hạt nhân không thể tìm thấy thiết bị `/dev/sda3` được thông báo là hệ thống tệp gốc.

Đáp án Bài tập Mở rộng

1. Trình tải khởi động sẽ hiển thị danh sách các hệ điều hành để chọn khi có nhiều hơn một hệ điều hành được cài đặt trên máy. Tuy nhiên, một hệ điều hành mới được cài đặt có thể sẽ ghi đè lên MBR của đĩa cứng, xóa giai đoạn đầu tiên của trình tải khởi động và khiến hệ điều hành khác không thể truy cập được. Tại sao điều này lại không xảy ra trên máy có phần sụn UEFI?

   Các máy UEFI không sử dụng MBR của đĩa cứng để lưu trữ giai đoạn đầu tiên của trình tải khởi động.

2. Một hậu quả phổ biến của việc cài đặt hạt nhân tùy chỉnh mà không cung cấp hình ảnh initramfs thích hợp là gì?

   Hệ thống tệp gốc có thể sẽ không truy cập được nếu loại của nó được biên dịch dưới dạng mô-đun nhân bên ngoài.

3. Nhật ký khởi tạo có thể dài đến hàng trăm dòng. Vì vậy, đầu ra của lệnh `dmesg` thường được chuyển thành lệnh máy nhắn tin — như lệnh `less` — để thuận tiện cho việc đọc. Tùy chọn `dmesg` nào sẽ tự động phân trang đầu ra của nó và loại bỏ nhu cầu sử dụng lệnh máy nhắn tin một cách rõ ràng?

   Các lệnh `dmesg -H` hoặc `dmesg --human` sẽ bật máy nhắn tin theo mặc định.

4. Ổ cứng chứa toàn bộ hệ thống tệp của máy ngoại tuyến đã bị xóa và gắn vào máy đang hoạt động dưới dạng ổ đĩa phụ. Giả sử điểm gắn kết của nó là `/mnt/hd` thì `journalctl` sẽ được sử dụng như thế nào để kiểm tra nội dung của các tệp nhật ký nằm ở `/mnt/hd/var/log/journal/`?

   Với lệnh `journalctl -D /mnt/hd/var/log/journal` hoặc `journalctl --directory=/mnt/hd/var/log/journal`.

101.3 Thay đổi mức chạy/mục tiêu khởi động và tắt hoặc khởi động lại hệ thống

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 101.3

Khối lượng

3

Các lĩnh vực kiến thức chính

- Đặt mục tiêu khởi động hoặc mức chạy mặc định.
- Thay đổi giữa các mức chạy / mục tiêu khởi động, bao gồm cả chế độ một người dùng.
- Tắt máy và khởi động lại từ dòng lệnh.
- Cảnh báo người dùng trước khi chuyển đổi mức chạy/mục tiêu khởi động hoặc các sự kiện hệ thống lớn khác.
- Chấm dứt tiến trình đúng cách.
- Kiến thức về acpid.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

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

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

101.3 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	101 Kiến trúc Hệ thống
Mục tiêu:	101.3 Thay đổi mức chạy/ mục tiêu khởi động và tắt hoặc khởi động lại Hệ thống
Bài:	1 trên 1

Giới thiệu

Một đặc điểm chung giữa các hệ điều hành tuân theo các nguyên tắc thiết kế Unix là việc sử dụng các tiến trình riêng biệt để kiểm soát các chức năng khác nhau của hệ thống. Các tiến trình này được gọi là *trình nền* (hay nói chung hơn là *các dịch vụ*). Chúng cũng chịu trách nhiệm về các tính năng mở rộng bên dưới hệ điều hành như dịch vụ ứng dụng mạng (máy chủ HTTP, chia sẻ tệp, email, v.v.), cơ sở dữ liệu, cấu hình theo yêu cầu, v.v. .Mặc dù Linux sử dụng hạt nhân nguyên khối, nhiều khía cạnh cấp thấp của hệ điều hành vẫn sẽ bị ảnh hưởng bởi trình nền như việc cân bằng tải và cấu hình tường lửa.

Việc trình nền nào nên hoạt động sẽ tùy thuộc vào mục đích của hệ thống. Tập hợp các trình nền đang hoạt động cũng có thể được sửa đổi trong thời gian chạy; vì vậy, các dịch vụ có thể được khởi động và dừng mà không cần phải khởi động lại toàn bộ hệ thống. Để giải quyết vấn đề này, mọi bản phân phối Linux lớn đều có cung cấp một số dạng tiện ích quản lý dịch vụ để kiểm soát hệ thống.

Các dịch vụ có thể được kiểm soát bởi các tệp lệnh vỏ hoặc bởi một chương trình và các tệp cấu hình hỗ trợ của nó. Phương pháp đầu tiên được triển khai theo tiêu chuẩn *SysVinit* hay còn được

gọi là *System V* hoặc chỉ *SysV*. Phương pháp thứ hai được triển khai bởi *systemd* và *Upstart*. Trước đây, các trình quản lý dịch vụ dựa trên SysV là được các bản phân phối Linux sử dụng nhiều nhất. Ngày nay, các trình quản lý dịch vụ dựa trên systemd thường phổ biến hơn trong hầu hết các bản phân phối Linux. Trình quản lý dịch vụ là chương trình đầu tiên được hạt nhân khởi chạy trong quá trình khởi động; vì vậy, PID (số nhận dạng tiến trình) của nó luôn là `1`.

SysVinit

Trình quản lý dịch vụ dựa trên tiêu chuẩn SysVinit sẽ cung cấp các tập hợp trạng thái hệ thống được xác định trước được gọi là *mức chạy* và các tệp tệp lệnh dịch vụ tương ứng của chúng sẽ được thực thi. Các mức chạy được đánh số từ `0` đến `6` và thường được chỉ định cho các mục đích sau:

Mức chạy số 0

Tắt hệ thống.

Mức chạy 1, s hoặc single

Chế độ một người dùng, không có mạng và các khả năng không cần thiết khác (chế độ bảo trì).

Mức chạy 2, 3 hoặc 4

Chế độ đa người dùng. Người dùng có thể đăng nhập bằng bảng điều khiển hoặc mạng. Mức chạy số 2 và 4 không được sử dụng thường xuyên.

Mức chạy số 5

Chế độ đa người dùng. Nó tương đương với mức 3 và có thêm đăng nhập chế độ đồ họa.

Mức chạy số 6

Khởi động lại hệ thống.

Chương trình chịu trách nhiệm quản lý các mức chạy và trình nền/tài nguyên liên quan là `/sbin/init`. Trong quá trình khởi tạo hệ thống, chương trình `init` sẽ xác định mức chạy được yêu cầu được chỉ định bởi tham số hạt nhân hoặc trong tệp `/etc/inittab` và tải các tệp lệnh liên quan được liệt kê ở đó cho mức chạy đã xác định. Mỗi mức chạy có thể có nhiều tệp dịch vụ liên quan, thường là các tệp lệnh trong thư mục `/etc/init.d/`. Vì không phải tất cả các mức chạy đều tương đương trong tất cả các bản phân phối của Linux nên chúng ta cũng có thể tìm thấy một mô tả ngắn về mục đích của mức chạy trong các bản phân phối dựa trên SysV.

Cú pháp của tệp `/etc/inittab` sẽ sử dụng định dạng này:

```
id:runlevels:action:process
```

`id` là tên chung có độ dài tối đa bốn ký tự được sử dụng để xác định mục nhập. Mục nhập `runlevels` là danh sách các số mức chạy để thực hiện một hành động cụ thể. Thuật ngữ `action` sẽ xác định cách thức `init` sẽ thực thi tiến trình được chỉ định bởi thuật ngữ `process`. Các hành động có sẵn là:

`boot`

Tiến trình sẽ được thực hiện trong quá trình khởi tạo hệ thống. Trường `runlevels` sẽ bị bỏ qua.

`bootwait`

Tiến trình sẽ được thực thi trong quá trình khởi tạo hệ thống và `init` sẽ đợi cho đến khi tiến trình kết thúc

`sysinit`

Tiến trình sẽ được thực hiện sau khi khởi tạo hệ thống dù mức chạy là gì. Trường `runlevels` sẽ bị bỏ qua.

`wait`

Tiến trình sẽ được thực thi cho các mức chạy nhất định và `init` sẽ đợi cho đến khi tiến trình kết thúc để tiếp tục.

`respawn`

Tiến trình sẽ được bắt đầu lại nếu bị chấm dứt.

`ctrlaltdel`

Tiến trình sẽ được thực thi khi quá trình init nhận được tín hiệu `SIGINT` được kích hoạt khi ta nhấn chuỗi phím `Ctrl` + `Alt` + `Del`.

Mức chạy mặc định — mức sẽ được chọn nếu không có mức nào khác được cung cấp làm tham số hạt nhân — cũng sẽ được xác định trong `/etc/inittab` trong mục nhập `id:x:initdefault`. `x` là số của mức chạy mặc định. Con số này sẽ không bao giờ là `0` hoặc `6` vì nó sẽ khiến hệ thống tắt hoặc khởi động lại ngay sau khi kết thúc quá trình khởi động. Một tệp `/etc/inittab` điển hình sẽ giống như dưới đây:

```
# 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
```

Lệnh `telinit q` nên được thực thi sau mỗi lần sửa đổi tệp `/etc/inittab`. Đối số `q` (hoặc `Q`) sẽ yêu cầu init tải lại cấu hình của nó. Bước này rất quan trọng trong việc tránh gián đoạn hệ thống do cấu hình không chính xác trong `/etc/inittab`.

Các tệp lệnh được `init` sử dụng để thiết lập từng mức chạy sẽ được lưu trữ trong thư mục `/etc/init.d/`. Mỗi mức chạy sẽ có một thư mục được liên kết trong `/etc/` được đặt tên là `/etc/rc0.d/`, `/etc/rc1.d/`, `/etc/rc2.d/`, v.v., với các tệp lệnh sẽ được thực thi khi mức chạy tương ứng bắt đầu. Vì cùng một tệp lệnh có thể được sử dụng bởi nhiều mức chạy khác nhau, các tệp trong các thư mục đó sẽ chỉ là các liên kết tượng trưng đến các tệp lệnh thực tế trong `/etc/init.d/`. Ngoài ra, chữ cái đầu tiên của tên tệp liên kết trong thư mục của mức chạy cho biết dịch vụ sẽ được bắt đầu hay kết thúc cho mức chạy tương ứng. Tên tệp của liên kết bắt đầu bằng chữ cái `K` (kill) xác định dịch vụ sẽ bị hủy khi nhập mức chạy. Nếu bắt đầu bằng chữ cái `S` (start), dịch vụ sẽ được bắt đầu khi vào mức chạy. Ví dụ: thư mục `/etc/rc1.d/` sẽ có nhiều liên kết đến các tệp lệnh mạng bắt đầu bằng chữ cái `K` vì mức chạy 1 là mức chạy một người dùng và không có kết nối mạng.

Lệnh `runlevel` sẽ hiển thị mức chạy hiện tại cho hệ thống. Lệnh `runlevel` sẽ hiển thị hai giá trị:

giá trị đầu tiên là mức chạy trước đó và giá trị thứ hai là mức chạy hiện tại:

```
$ runlevel
N 3
```

Chữ cái `N` trong đầu ra cho thấy mức chạy đã không thay đổi kể từ lần khởi động trước. Trong ví dụ này, `runlevel 3` là mức chạy hiện tại của hệ thống.

Chương trình `init` tương tự có thể được sử dụng để luân phiên giữa các mức chạy trong một hệ thống đang chạy mà không cần phải khởi động lại. Lệnh `telinit` cũng có thể được sử dụng để luân phiên giữa các mức chạy. Ví dụ: các lệnh `telinit 1`, `telinit s` hoặc `telinit S` sẽ chuyển hệ thống sang mức chạy 1.

systemd

Hiện tại, systemd là bộ công cụ được sử dụng rộng rãi nhất để quản lý tài nguyên và dịch vụ hệ thống (được systemd gọi là *các đơn vị*). Một đơn vị sẽ bao gồm tên, loại và tệp cấu hình tương ứng. Ví dụ: đơn vị cho tiến trình máy chủ *httpd* (như máy chủ web Apache) sẽ là `httpd.service` trên các bản phân phối dựa trên Red Hat và tệp cấu hình của nó cũng sẽ được gọi là `httpd.service` (trên các bản phân phối dựa trên Debian, đơn vị này là được đặt tên là `apache2.service`).

Có bảy loại đơn vị systemd riêng biệt:

`service`

Loại đơn vị phổ biến nhất, được dành cho các tài nguyên hệ thống đang hoạt động có thể được khởi tạo, gián đoạn và tải lại.

`socket`

Loại đơn vị ổ nối có thể là ổ nối hệ thống tệp hoặc ổ nối mạng. Tất cả các đơn vị ổ nối đều có một đơn vị dịch vụ tương ứng được tải khi ổ nối nhận được yêu cầu.

`device`

Một đơn vị thiết bị được liên kết với một thiết bị phần cứng xác định bởi hạt nhân. Một thiết bị sẽ chỉ được coi là một đơn vị systemd nếu có tồn tại quy tắc udev cho mục đích này. Một đơn vị thiết bị có thể được sử dụng để giải quyết các phần phụ thuộc cấu hình khi phát hiện các phần cứng nhất định (với điều kiện là các thuộc tính từ quy tắc udev có thể được sử dụng làm tham số cho đơn vị thiết bị).

`mount`

Đơn vị gắn kết là một định nghĩa điểm gắn kết trong hệ thống tệp, tương tự như một mục nhập

trong `/etc/fstab`.

`automount`

Một đơn vị automount cũng là một định nghĩa điểm gắn kết trong hệ thống tệp nhưng sẽ được gắn tự động. Mỗi đơn vị gắn kết tự động sẽ có một đơn vị gắn kết tương ứng. Đơn vị này sẽ được bắt đầu khi điểm gắn kết tự động được truy cập.

`target`

Một đơn vị mục tiêu là một nhóm các đơn vị khác được quản lý như một đơn vị.

`snapshot`

Đơn vị hình ảnh tức thời là một trạng thái đã lưu của trình quản lý systemd (không khả dụng trên mọi bản phân phối Linux).

Lệnh chính để điều khiển các đơn vị systemd là `systemctl`. Lệnh `systemctl` được sử dụng để thực thi tất cả các tác vụ liên quan đến kích hoạt, hủy kích hoạt, thực thi, gián đoạn, giám sát đơn vị, v.v. Chẳng hạn như đối với đơn vị hư cấu có tên `unit.service`, các hành động `systemctl` phổ biến nhất sẽ là:

`systemctl start unit.service`

Khởi động `unit`.

`systemctl stop unit.service`

Dừng `unit`.

`systemctl restart unit.service`

Khởi động lại `unit`.

`systemctl status unit.service`

Hiển thị trạng thái của `unit`, bao gồm cả việc nó có đang chạy hay không.

`systemctl is-active unit.service`

Hiển thị "hoạt động" (*active*) nếu `unit` đang chạy hoặc ngược lại.

`systemctl enable unit.service`

Kích hoạt `unit`, tức là `unit` sẽ tải trong quá trình khởi tạo hệ thống.

`systemctl disable unit.service`

`unit` sẽ không khởi động cùng hệ thống.

`systemctl is-enabled unit.service`

Xác minh xem `unit` có khởi động cùng hệ thống hay không. Câu trả lời sẽ được lưu trữ trong biến `$?`. Giá trị `0` cho biết `unit` sẽ khởi động cùng hệ thống và giá trị `1` cho biết rằng `unit` sẽ không khởi động cùng hệ thống.

NOTE

Các bản cài đặt mới hơn của systemd sẽ thực sự liệt kê cấu hình của một đơn vị cho thời gian khởi động. Ví dụ:

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

Nếu không có đơn vị nào khác có cùng tên tồn tại trong hệ thống thì hậu tố sau dấu chấm có thể bị loại bỏ. Ví dụ: nếu chỉ có một đơn vị `httpd` thuộc loại `service` thì chỉ cần `httpd` đã là đủ để làm thông số đơn vị cho `systemctl`.

Lệnh `systemctl` cũng có thể kiểm soát *các mục tiêu hệ thống*. Ví dụ: đơn vị `multi-user.target` sẽ kết hợp tất cả các đơn vị theo yêu cầu của môi trường hệ thống đa người dùng. Nó cũng tương tự như mức chạy số 3 trong một hệ thống sử dụng SysV.

Lệnh `systemctl isolate` sẽ luân phiên giữa các mục tiêu khác nhau. Vì vậy, để thay thế thủ công nhằm mục tiêu `multi-user`:

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

Có các mục tiêu (target) tương ứng cho các mức chạy SysV bắt đầu từ `runlevel0.target` cho đến `runlevel6.target`. Tuy nhiên, systemd không sử dụng tệp `/etc/inittab`. Để thay đổi mục tiêu hệ thống mặc định, ta có thể thêm tùy chọn `systemd.unit` vào danh sách tham số hạt nhân. Ví dụ: để sử dụng `multi-user.target` làm mục tiêu tiêu chuẩn, tham số hạt nhân sẽ phải là `systemd.unit=multi-user.target`. Tất cả các tham số hạt nhân có thể được duy trì bằng cách thay đổi cấu hình trình tải khởi động.

Một cách khác để thay đổi mục tiêu mặc định là sửa đổi liên kết tượng trưng `/etc/systemd/system/default.target` để nó trỏ đến mục tiêu mong muốn. Việc xác định lại liên kết có thể được thực hiện bằng chính lệnh `systemctl`:

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

Tương tự, ta có thể xác định mục tiêu khởi động mặc định của hệ thống bằng lệnh sau:

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

Tương tự như các hệ thống sử dụng SysV, mục tiêu mặc định không bao giờ nên trỏ đến `shutdown.target` vì nó tương ứng với mức chạy 0 (tắt máy).

Ta có thể tìm thấy các tệp cấu hình được liên kết với mọi đơn vị trong thư mục `/lib/systemd/system/`. Lệnh `systemctl list-unit-files` sẽ liệt kê tất cả các đơn vị khả dụng và hiển thị nếu chúng được kích hoạt để khởi động khi hệ thống khởi động. Tùy chọn `--type` sẽ chỉ chọn các đơn vị cho một loại nhất định (như trong `systemctl list-unit-files --type=service` và `systemctl list-unit-files --type=target`).

Các đơn vị đang hoạt động hoặc các đơn vị đã hoạt động trong phiên hệ thống hiện tại có thể được liệt kê bằng lệnh `systemctl list-units`. Giống như tùy chọn `list-unit-files`, lệnh `systemctl list-units --type=service` sẽ chỉ chọn các đơn vị thuộc loại `service` và lệnh `systemctl list-units --type=target` sẽ chỉ chọn các đơn vị thuộc loại `target`.

systemd cũng chịu trách nhiệm kích hoạt và phản hồi các sự kiện liên quan đến nguồn điện. Lệnh `systemctl suspend` sẽ đặt hệ thống ở chế độ năng lượng thấp và giữ dữ liệu hiện tại trong bộ nhớ. Lệnh `systemctl hibernate` sẽ sao chép tất cả dữ liệu bộ nhớ vào đĩa để trạng thái hiện tại của hệ thống có thể được khôi phục sau khi tắt nguồn. Các hành động liên quan đến các sự kiện như vậy sẽ được xác định trong tệp `/etc/systemd/logind.conf` hoặc trong các tệp riêng biệt bên trong thư mục `/etc/systemd/logind.conf.d/`. Tuy nhiên, tính năng systemd này chỉ có thể được sử dụng khi không có trình quản lý năng lượng nào khác đang chạy trong hệ thống, chẳng hạn như trình nền `acpid`. Trình nền `acpid` là trình quản lý năng lượng chính cho Linux và sẽ cho phép điều chỉnh tốt hơn các hành động sau các sự kiện liên quan đến năng lượng như đóng nắp máy tính xách tay, pin yếu hoặc mức sạc pin.

Upstart

Các tệp lệnh khởi tạo được Upstart sử dụng sẽ nằm trong thư mục `/etc/init/`. Các dịch vụ hệ thống có thể được liệt kê bằng lệnh `initctl list`. Lệnh này cũng hiển thị trạng thái hiện tại của các dịch vụ và số PID của chúng (nếu có).

```
# 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
```

Mọi hành động Upstart đều có lệnh độc lập của riêng nó. Ví dụ: lệnh `start` có thể được sử dụng để khởi tạo cửa sổ dòng lệnh ảo thứ sáu:

```
# start tty6
```

Trạng thái hiện tại của tài nguyên có thể được xác minh bằng lệnh `status`:

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

Và việc gián đoạn một dịch vụ có thể được thực hiện bằng lệnh `stop`:

```
# stop tty6
```

Upstart không sử dụng tệp `/etc/inittab` để xác định các mức chạy, nhưng các lệnh kế thừa `runlevel` và `telinit` vẫn có thể được sử dụng để xác minh và luân phiên giữa các mức chạy.

NOTE Upstart được phát triển cho bản phân phối Ubuntu Linux để giúp tạo điều kiện khởi động song song các tiến trình. Ubuntu đã ngừng sử dụng Upstart từ năm 2015 và chuyển từ Upstart sang systemd.

Tắt máy và Khởi động lại

Một lệnh rất truyền thống được sử dụng để tắt hoặc khởi động lại hệ thống được gọi là `shutdown`. Lệnh `shutdown` sẽ bổ sung các chức năng cho tiến trình tắt nguồn: lệnh này sẽ tự động thông báo cho tất cả những người dùng đã đăng nhập bằng thông báo cảnh báo trong các phiên vỏ của họ và

các lần đăng nhập mới sẽ bị ngăn chặn. Lệnh `shutdown` đóng vai trò trung gian cho các quy trình SysV hoặc systemd, nghĩa là nó sẽ thực thi hành động được yêu cầu bằng cách gọi hành động tương ứng trong trình quản lý dịch vụ đã được hệ thống thông qua.

Sau khi `shutdown` được thực thi, tất cả các tiến trình đều sẽ nhận được lần lượt tín hiệu `SIGTERM` và `SIGKILL`. Sau đó, hệ thống sẽ tắt hoặc thay đổi mức chạy của nó. Theo mặc định, khi cả hai tùy chọn `-h` hoặc `-r` đều không được sử dụng, hệ thống sẽ chuyển sang mức chạy 1, nghĩa là chế độ một người dùng. Để thay đổi các tùy chọn mặc định cho `shutdown`, lệnh phải được thực hiện theo cú pháp sau:

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

Chỉ có tham số `time` là bắt buộc. Tham số `time` sẽ xác định khi nào hành động được yêu cầu được thực thi. Nó sẽ chấp nhận các định dạng sau:

`hh:mm`

Định dạng này chỉ định thời gian thực hiện là giờ (h) và phút (m).

`+m`

Định dạng này chỉ định số phút chờ trước khi thực thi.

`now` or `+0`

Định dạng này xác định việc thực thi ngay lập tức.

Tham số `message` là văn bản cảnh báo được gửi tới tất cả các phiên cửa sổ dòng lệnh của những người dùng đã đăng nhập.

Việc triển khai SysV cho phép giới hạn người dùng có thể khởi động lại máy bằng cách nhấn `Ctrl` + `Alt` + `Del`. Điều này có thể thực hiện được bằng cách đặt tùy chọn `-a` cho lệnh `shutdown` ở dòng liên quan đến `ctrlaltdel` trong tệp `/etc/inittab`. Bằng cách này, chỉ những người dùng có tên người dùng nằm trong tệp `/etc/shutdown.allow` mới có thể khởi động lại hệ thống bằng tổ hợp phím `Ctrl` + `Alt` + `Del`.

Lệnh `systemctl` cũng có thể được sử dụng để tắt hoặc khởi động lại máy trong các hệ thống sử dụng systemd. Để khởi động lại hệ thống, ta nên sử dụng lệnh `systemctl reboot`. Để tắt hệ thống, ta nên sử dụng lệnh `systemctl poweroff`. Cả hai lệnh đều yêu cầu quyền gốc để chạy vì người dùng thông thường không thể thực hiện các tiến trình đó.

NOTE

Một số bản phân phối Linux sẽ liên kết `poweroff` và `reboot` với `systemctl` dưới dạng các lệnh riêng lẻ. Ví dụ:

```
$ 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
```

Không phải hoạt động bảo trì nào cũng sẽ yêu cầu tắt hoặc khởi động lại hệ thống. Tuy nhiên, khi cần thay đổi trạng thái của hệ thống sang chế độ một người dùng, điều quan trọng là phải cảnh báo những người dùng đã đăng nhập để họ không bị tổn hại do các hoạt động của họ sẽ bị chấm dứt đột ngột.

Tương tự như cách lệnh `shutdown` thực hiện khi tắt nguồn hoặc khởi động lại hệ thống, lệnh `wall` có thể gửi thông báo đến các phiên cửa sổ dòng lệnh của tất cả người dùng đã đăng nhập. Để làm như vậy, quản trị viên hệ thống chỉ cần cung cấp tệp hoặc viết trực tiếp thông báo dưới dạng tham số cho lệnh `wall`.

Bài tập Hướng dẫn

1. Làm thế nào để lệnh `telinit` có thể được sử dụng để khởi động lại hệ thống?

2. Điều gì sẽ xảy ra với các dịch vụ liên quan đến tệp `/etc/rc1.d/K90network` khi hệ thống tiến vào mức chạy 1?

3. Bằng cách sử dụng lệnh `systemctl`, làm cách nào để người dùng có thể xác minh xem đơn vị `sshd.service` có đang chạy hay không?

4. Trong một hệ thống dựa trên systemd, lệnh nào phải được thực thi để cho phép kích hoạt đơn vị `sshd.service` trong quá trình khởi tạo hệ thống?

Bài tập Mở rộng

1. Trong hệ thống dựa trên SysV, giả sử mức chạy mặc định được xác định trong `/etc/inittab` là 3 nhưng hệ thống luôn bắt đầu ở mức chạy 1. Nguyên nhân có thể là gì?

2. Mặc dù có thể tìm thấy tệp `/sbin/init` trong các hệ thống dựa trên systemd nhưng nó chỉ là một liên kết tượng trưng đến một tệp thực thi khác. Trong các hệ thống như vậy, tệp được trỏ bởi `/sbin/init` là gì?

3. Làm cách nào để xác minh mục tiêu hệ thống mặc định trong hệ thống dựa trên systemd?

4. Làm cách nào để hủy khởi động lại hệ thống đã lên lịch bằng cách sử dụng lệnh `shutdown`?

Tóm tắt

Nội dung bài học này bao gồm các tiện ích chính được sử dụng làm trình quản lý dịch vụ của các bản phân phối Linux. Mỗi tiện ích như SysVinit, systemd và Upstart đều có cách tiếp cận riêng để kiểm soát các dịch vụ hệ thống và trạng thái hệ thống. Bài học đã đi qua các chủ đề sau:

- Dịch vụ hệ thống là gì và vai trò của chúng trong hệ điều hành.
- Khái niệm và cách sử dụng cơ bản của các lệnh SysVinit, systemd và Upstart.
- Cách chính xác để khởi động, dừng và khởi động lại các dịch vụ hệ thống và chính hệ thống.

Các lệnh và tiến trình đã được nhắc tới là:

- Các lệnh và tệp liên quan đến SysVinit, như `init`, `/etc/inittab` và `telinit`.
- Lệnh systemd chính: `systemctl`.
- Các lệnh khởi động: `initctl`, `status`, `start`, `stop`.
- Các lệnh quản lý năng lượng truyền thống như `shutdown` và lệnh `wall`.

Đáp án Bài tập Hướng dẫn

1. Làm thế nào để lệnh `telinit` có thể được sử dụng để khởi động lại hệ thống?

   Lệnh `telinit 6` sẽ chuyển sang mức chạy 6, tức là khởi động lại hệ thống.

2. Điều gì sẽ xảy ra với các dịch vụ liên quan đến tệp `/etc/rc1.d/K90network` khi hệ thống tiến vào mức chạy 1?

   Do có chữ `K` ở đầu tên tệp nên các dịch vụ liên quan sẽ bị dừng.

3. Bằng cách sử dụng lệnh `systemctl`, làm cách nào để người dùng có thể xác minh xem đơn vị `sshd.service` có đang chạy hay không?

   Với lệnh `systemctl status sshd.service` hoặc `systemctl is-active sshd.service`.

4. Trong một hệ thống dựa trên systemd, lệnh nào phải được thực thi để cho phép kích hoạt đơn vị `sshd.service` trong quá trình khởi tạo hệ thống?

   Lệnh `systemctl enable sshd.service` được thực thi bởi siêu người dùng.

Đáp án Bài tập Mở rộng

1. Trong hệ thống dựa trên SysV, giả sử mức chạy mặc định được xác định trong `/etc/inittab` là 3 nhưng hệ thống luôn bắt đầu ở mức chạy 1. Nguyên nhân có thể là gì?

   Các tham số `1` hoặc `S` có thể có trong danh sách tham số của hạt nhân.

2. Mặc dù có thể tìm thấy tệp `/sbin/init` trong các hệ thống dựa trên systemd nhưng nó chỉ là một liên kết tượng trưng đến một tệp thực thi khác. Trong các hệ thống như vậy, tệp được trỏ bởi `/sbin/init` là gì?

   Nhị phân systemd chính: `/lib/systemd/systemd`.

3. Làm cách nào để xác minh mục tiêu hệ thống mặc định trong hệ thống dựa trên systemd?

   Liên kết tượng trưng `/etc/systemd/system/default.target` sẽ trỏ đến tệp đơn vị được xác định là mục tiêu mặc định. Lệnh `systemctl get-default` cũng có thể được sử dụng.

4. Làm cách nào để hủy khởi động lại hệ thống đã lên lịch bằng cách sử dụng lệnh `shutdown`?

   Nên sử dụng lệnh `shutdown -c`.

Linux Professional Institute

Chủ đề 102: Cài đặt Linux và Quản lý Gói

102.1 Thiết kế Bố cục Ổ đĩa cứng

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 102.1

Khối lượng

2

Các lĩnh vực kiến thức chính

- Phân bổ hệ thống tệp và không gian hoán đổi cho các phân vùng hoặc đĩa riêng biệt.
- Điều chỉnh thiết kế theo mục đích sử dụng của hệ thống.
- Đảm bảo phân vùng /boot tuân thủ các yêu cầu về kiến trúc phần cứng để khởi động.
- Kiến thức về các tính năng cơ bản của LVM.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

- `/` (root) hệ thống tệp
- `/var` hệ thống tệp
- `/home` hệ thống tệp
- `/boot` hệ thống tệp
- EFI System Partition (ESP)
- không gian hoán đổi
- điểm gắn kết
- phân vùng

102.1 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	102 Cài đặt Linux và Quản lý Gói
Mục tiêu:	102.1 Thiết kế Bố cục Ổ đĩa cứng
Bài:	1 trên 1

Giới thiệu

Để đạt được mục tiêu của bài học này, chúng ta cần hiểu về mối quan hệ giữa *ổ đĩa cứng (disk)*, *phân vùng (partion)*, *hệ thống tệp (filesystem)* và *ổ phân vùng (volume)*.

Hãy coi đĩa (hoặc *thiết bị lưu trữ* vì các thiết bị hiện đại hoàn toàn không chứa bất kỳ một "đĩa" nào) như là một "vùng chứa vật lý" dành cho dữ liệu của bạn.

Trước khi một đĩa có thể được máy tính sử dụng, nó sẽ cần được phân vùng. Phân vùng là một tập hợp con logic của ổ đĩa vật lý, giống như một "hàng rào" logic. Phân vùng là một phương thức để "ngăn cách" các thông tin được lưu trữ trên đĩa, chẳng hạn như ngăn cách dữ liệu hệ điều hành khỏi dữ liệu người dùng.

Mỗi đĩa cần ít nhất một phân vùng, nhưng nó cũng có thể có nhiều phân vùng nếu cần và thông tin về chúng sẽ được lưu trữ trong bảng phân vùng. Bảng này bao gồm thông tin về các khu vực đầu tiên và cuối cùng của phân vùng, loại của nó cũng như các chi tiết khác về từng phân vùng.

Bên trong mỗi phân vùng sẽ có một hệ thống tệp. Hệ thống tệp mô tả cách thông tin thực sự được lưu trữ trên đĩa. Thông tin này bao gồm cách tổ chức các thư mục, mối quan hệ giữa chúng là gì, dữ liệu cho mỗi tệp nằm ở đâu, v.v.

Các phân vùng không thể được mở rộng trên nhiều đĩa. Nhưng bằng cách sử dụng *Trình Quản lý Ổ phân vùng Logic* (LVM), nhiều phân vùng có thể được kết hợp (thậm chí là kết hợp giữa các đĩa với nhau) để tạo thành một ổ đĩa logic duy nhất.

Các ổ phân vùng logic sẽ trừu tượng hóa các hạn chế của thiết bị vật lý và cho phép ta làm việc với "vùng" dung lượng đĩa có thể được kết hợp hoặc phân phối theo cách linh hoạt hơn nhiều so với các phân vùng truyền thống. LVM rất hữu ích trong các trường hợp cần thêm nhiều dung lượng vào một phân vùng mà không cần di chuyển dữ liệu sang một thiết bị lớn hơn.

Trong mục tiêu này, chúng ta sẽ tìm hiểu cách thiết kế sơ đồ phân vùng đĩa cho hệ thống Linux, phân bổ hệ thống tệp và hoán đổi dung lượng cho các phân vùng hoặc đĩa riêng biệt khi cần.

Cách *tạo* và *quản lý* các phân vùng và hệ thống tệp sẽ được thảo luận trong các bài học tới. Trong mục tiêu này, chúng ta sẽ thảo luận tổng quan về LVM (phần giải thích chi tiết sẽ nằm ngoài phạm vi của mục tiêu).

Điểm Gắn kết

Trước khi một hệ thống tệp có thể được truy cập trên Linux, nó cần phải được *gắn kết*. Việc gắn hệ thống tệp vào một điểm cụ thể trong cây thư mục của hệ thống được gọi là *điểm gắn kết*.

Khi được gắn, nội dung của hệ thống tệp sẽ có sẵn trong điểm gắn kết. Ví dụ: hãy tưởng tượng bạn có một phân vùng chứa dữ liệu cá nhân của người dùng (thư mục chính của họ) có chứa các thư mục `/john`, `/jack` và `/carol`. Khi được gắn trong `/home`, nội dung của các thư mục đó sẽ có sẵn trong `/home/john`, `/home/jack` và `/home/carol`.

Điểm gắn kết phải tồn tại trước khi gắn hệ thống tệp. Ta không thể gắn phân vùng trong `/mnt/userdata` nếu thư mục này không tồn tại. Tuy nhiên, nếu thư mục này tồn tại và chứa các tệp, các tệp đó sẽ không khả dụng cho đến khi ta ngắt kết nối hệ thống tệp. Nếu liệt kê nội dung của thư mục, ta sẽ thấy các đơn vị được gắn kết là các tệp được lưu trữ trên hệ thống tệp chứ không phải nội dung gốc của thư mục.

Hệ thống tệp có thể được gắn ở bất cứ nơi nào ta muốn. Tuy nhiên, có một số thông lệ hữu ích nên được tuân theo để giúp việc quản trị hệ thống trở nên dễ dàng hơn.

Theo truyền thống, `/mnt` là thư mục mà tất cả các thiết bị ngoại vi sẽ được gắn vào và một số *điểm neo* được cấu hình sẵn cho các thiết bị phổ biến như đĩa CD-ROM (`/mnt/cdrom`) và đĩa mềm (`/mnt/floppy`) sẽ tồn tại bên dưới nó.

Thư mục này đã được thay thế bởi `/media` - hiện là điểm gắn kết mặc định cho bất kỳ phương tiện nào mà người dùng có thể tháo rời (ví dụ: ổ đĩa ngoại vi, ổ flash USB, đầu đọc thẻ nhớ, đĩa quang, v.v.) được kết nối với hệ thống.

Trên hầu hết các bản phân phối Linux và môi trường máy tính để bàn hiện đại, các thiết bị di động sẽ được tự động gắn vào `/media/USER/LABEL` khi được kết nối với hệ thống, trong đó, `USER` là tên người dùng và `LABEL` là nhãn thiết bị. Ví dụ: ổ flash USB có nhãn `FlashDrive` được kết nối bởi người dùng `john` sẽ được gắn trong `/media/john/FlashDrive/`. Cách để xử lý vấn đề này sẽ khác nhau tùy thuộc vào môi trường máy tính để bàn.

Từ đó, chúng ta có thể hiểu rằng bất cứ khi nào cần gắn hệ thống tệp *thủ công*, ta nên gắn hệ thống đó trong `/mnt`. Các lệnh cụ thể để kiểm soát việc gắn kết và ngắt kết nối các hệ thống tệp trong Linux sẽ được thảo luận trong một bài học khác.

Giữ mọi thứ tách biệt

Có một số thư mục trong Linux mà chúng ta nên cân nhắc để giữ trên các phân vùng riêng biệt. Có nhiều lý do cho điều này, ví dụ như bằng cách giữ các tệp liên quan đến trình tải khởi động (được lưu trữ trên `/boot`) trên *phân vùng khởi động*, ta sẽ đảm bảo được hệ thống của mình vẫn có thể khởi động trong trường hợp hệ thống tệp gốc gặp sự cố.

Việc giữ các thư mục cá nhân của người dùng (trong `/home`) trên một phân vùng riêng sẽ giúp việc cài đặt lại hệ thống trở nên dễ dàng hơn mà không có nguy cơ vô tình chạm tới dữ liệu người dùng. Việc lưu giữ dữ liệu liên quan đến một trang web hoặc máy chủ cơ sở dữ liệu (thường dưới `/var`) trên một phân vùng riêng (hoặc thậm chí là một đĩa riêng) sẽ giúp việc quản trị hệ thống trở nên dễ dàng hơn nếu ta cần thêm nhiều dung lượng đĩa hơn cho các trường hợp sử dụng đó.

Thậm chí có thể có những lý do liên quan đến hiệu suất khiến chúng ta phải cân nhắc việc giữ một số thư mục trên các phân vùng riêng biệt. Ta có thể muốn giữ hệ thống tệp gốc (`/`) trên ổ SSD tốc độ cao và các thư mục lớn hơn như `/home` và `/var` trên các ổ đĩa cứng chậm hơn, cung cấp nhiều dung lượng hơn với mức tiêu hao thấp.

Phân vùng Khởi động (`/boot`)

Phân vùng khởi động có chứa các tệp được trình tải khởi động sử dụng để tải hệ điều hành. Trên các hệ thống Linux, trình tải khởi động thường là GRUB2 hoặc là GRUB Legacy trên các hệ thống cũ hơn. Phân vùng này thường được gắn trong `/boot` và các tệp của nó sẽ được lưu trữ trong `/boot/grub`.

Về mặt kỹ thuật, hệ thông không cần tới phân vùng khởi động vì trong hầu hết các trường hợp, GRUB có thể gắn phân vùng gốc (`/`) và tải các tệp từ một thư mục `/boot` riêng biệt.

Tuy nhiên, có thể hệ thống sẽ cần một phân vùng khởi động riêng để đảm bảo an toàn (đảm bảo hệ thống sẽ khởi động ngay cả trong trường hợp hệ thống tệp gốc gặp sự cố) hoặc nếu người dùng muốn sử dụng một hệ thống tệp mà trình tải khởi động không thể hiểu được trong phân vùng gốc,

hoặc nếu nó sử dụng một phương pháp mã hóa hoặc nén không được hỗ trợ.

Phân vùng khởi động thường là phân vùng đầu tiên trên đĩa. Điều này là do BIOS PC ban đầu của IBM xử lý các đĩa bằng cách sử dụng CHS, tức *xilanh* (cylinders), *đầu* (heads) và *ngành* (sectors), với tối đa 1024 xi lanh, 256 đầu và 63 ngành, dẫn đến kích thước đĩa tối đa là 528 MB (504 MB trong MS-DOS). Điều này có nghĩa là mọi thứ vượt qua mốc này sẽ không thể truy cập được trên các hệ thống cũ trừ khi một sơ đồ địa chỉ đĩa khác (như *Định Địa chỉ Khối Logic* - LBA) đã được sử dụng.

Vì vậy, để tương thích tối đa, phân vùng khởi động thường nằm ở đầu đĩa và kết thúc trước xi lanh 1024 (528 MB) để đảm bảo rằng dù thế nào đi chăng nữa, máy vẫn sẽ luôn tải được hạt nhân.

Vì phân vùng khởi động chỉ lưu trữ các tệp cần thiết cho trình tải khởi động (tức đĩa RAM ban đầu và hình ảnh hạt nhân) nên nó có thể có kích thước khá nhỏ theo tiêu chuẩn ngày nay. Một kích thước hợp lý là khoảng 300 MB.

Phân vùng hệ thống EFI (ESP)

Phân vùng hệ thống EFI (ESP) được sử dụng bởi các máy dựa trên *Giao diện Phần Sụn Mở rộng Hợp nhất* (UEFI) để lưu trữ các trình tải khởi động và hình ảnh hạt nhân cho các hệ điều hành được cài đặt.

Phân vùng này được định dạng trong hệ thống tệp dựa trên FAT. Trên đĩa được phân vùng bằng Bảng phân vùng GUID, nó có mã định danh duy nhất trên toàn cầu là `C12A7328-F81F-11D2-BA4B-00A0C93EC93B`. Nếu đĩa được định dạng theo sơ đồ phân vùng MBR, ID phân vùng sẽ là `0xEF`.

Trên các máy chạy Microsoft Windows, phân vùng này thường là phân vùng đầu tiên trên đĩa (mặc dù điều này là không bắt buộc). ESP sẽ được hệ điều hành tạo (hoặc di dời tới) khi cài đặt và được gắn trong `/boot/efi` trên hệ thống Linux.

Phân vùng `/home`

Mỗi người dùng trong hệ thống đều có một thư mục chính để lưu trữ các tệp và tùy chọn cá nhân, và hầu hết chúng đều nằm trong `/home`. Thông thường, thư mục chính sẽ giống với tên người dùng; vì vậy, người dùng John sẽ có thư mục chính là `/home/john`.

Tuy nhiên thì vẫn có những trường hợp ngoại lệ. Ví dụ: thư mục chính cho siêu người dùng là `/root` và một số dịch vụ hệ thống sẽ có thể liên kết người dùng với các thư mục chính ở nơi khác.

Không có quy tắc nào để xác định kích thước của phân vùng cho thư mục `/home` (phân vùng chính). Chúng ta phải tính đến số lượng người dùng trong hệ thống và cách nó sẽ được sử dụng.

Chẳng hạn như một người dùng chỉ duyệt web và xử lý văn bản sẽ cần ít dung lượng hơn một người làm công việc chỉnh sửa video.

Dữ liệu Biến (`/var`)

Thư mục này chứa "dữ liệu biến" hoặc các tệp và thư mục mà hệ thống phải có khả năng ghi vào trong quá trình hoạt động. Điều này bao gồm nhật ký hệ thống (trong `/var/log`), tệp tạm thời (`/var/tmp`) và dữ liệu ứng dụng được lưu trong bộ nhớ đệm (trong `/var/cache`).

`/var/www/html` cũng là thư mục mặc định cho các tệp dữ liệu cho Máy chủ Web Apache và `/var/lib/mysql` là vị trí mặc định cho các tệp cơ sở dữ liệu cho máy chủ MySQL. Tuy nhiên, cả hai thư mục này đều có thể thay đổi được.

Một lý do chính đáng để đặt `/var` vào một phân vùng riêng là tính ổn định của nó. Có nhiều ứng dụng và tiến trình ghi vào `/var` và các thư mục con như `/var/log` hoặc `/var/tmp`. Một tiến trình hoạt động sai có thể ghi dữ liệu cho đến khi không còn dung lượng trống trên hệ thống tệp.

Nếu `/var` nằm dưới `/`, điều này có thể gây ra tình trạng loạn hạt nhân và hỏng hệ thống tệp, gây ra các tình huống khó khắc phục. Nhưng nếu `/var` được giữ dưới một phân vùng riêng thì hệ thống tệp gốc sẽ không bị ảnh hưởng.

Giống như trong `/home`, không có quy tắc chung nào để xác định kích thước của phân vùng cho `/var` vì nó sẽ thay đổi theo cách hệ thống được sử dụng. Trong hệ thống home thì có thể sẽ chỉ mất vài gigabyte. Nhưng trên cơ sở dữ liệu hoặc máy chủ web thì có thể sẽ cần nhiều dung lượng hơn. Trong những trường hợp như vậy, sẽ rất hữu ích khi đặt `/var` trên một phân vùng trên một đĩa khác với phân vùng gốc để tăng thêm một lớp bảo vệ chống lại lỗi đĩa vật lý.

Hoán đổi

Phân vùng hoán đổi được sử dụng để hoán đổi các trang bộ nhớ từ RAM sang đĩa khi cần. Phân vùng này cần phải thuộc một loại cụ thể và được thiết lập với tiện ích phù hợp có tên là `mkswap` trước khi có thể sử dụng được.

Phân vùng hoán đổi không thể được gắn kết như các phân vùng khác, nghĩa là ta không thể truy cập và xem nội dung của nó như một thư mục bình trường.

Một hệ thống có thể có nhiều phân vùng hoán đổi (mặc dù điều này không phổ biến) và Linux cũng hỗ trợ sử dụng *các tệp* hoán đổi thay vì các phân vùng. Điều này có thể sẽ hữu ích trong việc tăng nhanh dung lượng hoán đổi khi cần.

Kích thước của phân vùng trao đổi là một vấn đề gây tranh cãi. Quy tắc cũ từ những ngày đầu của Linux ("gấp đôi dung lượng RAM") có thể đã không còn được áp dụng nữa, tùy thuộc vào cách hệ

thống đang được sử dụng và lượng RAM vật lý được cài đặt.

Trên tài liệu dành cho Red Hat Enterprise Linux 7, Red Hat khuyến nghị như sau:

Dung lượng RAM	**Kích thước hoán đổi được đề xuất**	**Kích thước hoán đổi được đề xuất với Chế độ Ngủ đông**
< 2 GB RAM	Gấp đôi dung lượng RAM	Gấp 3 lần dung lượng RAM
RAM 2-8 GB	Bằng dung lượng RAM	Gấp đôi dung lượng RAM
RAM 8-64 GB	Ít nhất 4 GB	Gấp 1,5 lần dung lượng RAM
> 64 GB RAM	Ít nhất 4 GB	Không được khuyến khích

Tất nhiên dung lượng hoán đổi có thể phụ thuộc vào khối lượng công việc. Nếu máy đang chạy một dịch vụ quan trọng (chẳng hạn như cơ sở dữ liệu, web hoặc máy chủ SAP), ta nên kiểm tra tài liệu về các dịch vụ này (hoặc về nhà cung cấp phần mềm) để biết dung lượng được khuyến nghị.

NOTE Để biết thêm về cách tạo và kích hoạt phân vùng hoán đổi và tệp hoán đổi, hãy xem Mục tiêu 104.1 của LPIC-1.

LVM

Chúng ta đã thảo luận về cách các đĩa được tổ chức thành một hoặc nhiều phân vùng với mỗi phân vùng chứa một hệ thống tệp mô tả cách các tệp và siêu dữ liệu liên quan được lưu trữ. Một trong những nhược điểm của việc phân vùng là người quản trị hệ thống phải quyết định trước cách thức phân phối dung lượng đĩa khả dụng trên thiết bị. Điều này có thể đưa ra một số thách thức sau này nếu một phân vùng yêu cầu nhiều dung lượng hơn dự kiến ban đầu. Tất nhiên, các phân vùng có thể được thay đổi kích thước, nhưng điều này có thể không thực hiện được trong một số trường hợp (ví dụ như không có dung lượng trống trên đĩa).

Trình Quản lý Ổ phân vùng Logic (LVM - Logical Volume Management) là một dạng ảo hóa lưu trữ cung cấp cho quản trị viên hệ thống một cách tiếp cận linh hoạt hơn để quản lý dung lượng đĩa so với phương thức phân vùng truyền thống. Mục tiêu của LVM là tạo thuận lợi cho việc quản lý nhu cầu lưu trữ của người dùng cuối. Đơn vị cơ bản của nó là *Ổ phân vùng Vật lý* (PV), tức là một thiết bị khối trên hệ thống (như phân vùng đĩa hoặc mảng RAID).

Các PV được nhóm thành *các Nhóm Ổ đĩa* (VG) sẽ trừu tượng hóa các thiết bị cơ bản và được xem như một thiết bị logic duy nhất với dung lượng lưu trữ kết hợp của các PV thành phần.

Mỗi ổ phân vùng trong một Nhóm Ổ phân vùng được chia nhỏ thành các phần có kích thước cố định được gọi là *mức phạm vi* (extents). Mức phạm vi trên PV được gọi là *Mức phạm vi Vật lý* (PE), trong khi mức phạm vi trên Ổ phân vùng Logic là *Mức phạm vi Logic* (LE). Nói chung, mỗi Mức

phạm vi logic đều được ánh xạ tới Mức phạm vi vật lý, nhưng điều này có thể thay đổi nếu các tính năng như phản chiếu đĩa được sử dụng.

Các Nhóm Ổ đĩa có thể được chia nhỏ thành các Ổ đĩa Logic (LV) có chức năng hoạt động theo cách tương tự như các phân vùng nhưng linh hoạt hơn.

Kích thước của một Ổ phân vùng logic (như được chỉ định trong quá trình tạo) trên thực tế được xác định bằng kích thước của mức phạm vi vật lý (4 MB theo mặc định) nhân với số lượng mức phạm vi trên ổ phân vùng. Từ đó, ta có thể dễ dàng hiểu rằng để phát triển một đơn vị như Ổ phân vùng Logic, tất cả những gì quản trị viên hệ thống phải làm là thêm nhiều mức phạm vi hơn từ vùng có sẵn trong Nhóm Ổ phân vùng. Tương tự như vậy, các mức phạm vi cũng có thể được loại bỏ để thu nhỏ LV.

Sau khi một Ổ phân vùng Logic được tạo, hệ điều hành sẽ xem nó như một thiết bị khối thông thường. Một thiết bị sẽ được tạo trong `/dev` và được đặt tên là `/dev/VGNAME/LVNAME`, trong đó, `VGNAME` là tên của Nhóm Ổ phân vùng và `LVNAME` là tên của Ổ phân vùng Logic.

Các thiết bị này có thể được định dạng bằng hệ thống tệp mong muốn bằng cách sử dụng các tiện ích tiêu chuẩn (chẳng hạn như `mkfs.ext4`) và được gắn kết bằng các phương pháp thông thường, có thể là theo cách thủ công bằng lệnh `mount` hoặc tự động bằng cách thêm chúng vào tệp `/etc/fstab `.

Bài tập Hướng dẫn

1. Trên các hệ thống Linux, các tệp cho trình tải khởi động GRUB được lưu trữ ở đâu?

2. Phân vùng khởi động nên kết thúc ở đâu để đảm bảo rằng PC luôn có thể tải được hạt nhân?

3. Phân vùng EFI thường được gắn ở đâu?

4. Khi gắn thủ công một hệ thống tệp, nó thường được gắn vào thư mục nào?

Bài tập Mở rộng

1. Đơn vị nhỏ nhất bên trong một Nhóm Ổ đĩa là gì?

2. Kích thước của Ổ đĩa Logic được xác định như thế nào?

3. Trên một đĩa được định dạng bằng sơ đồ phân vùng MBR, ID của Phân vùng hệ thống EFI là gì?

4. Bên cạnh các phân vùng hoán đổi, làm thế nào để có thể nhanh chóng tăng dung lượng hoán đổi trên hệ thống Linux?

Tóm tắt

Trong bài học này, chúng ta đã tìm hiểu về phân vùng và thư mục nào thường được giữ trong các phân vùng riêng biệt và tại sao điều này được thực hiện. Ngoài ra, chúng ta đã thảo luận tổng quan về LVM (Trình Quản lý Ổ phân vùng Logic) và việc nó có thể cung cấp một cách linh hoạt hơn để phân bổ dữ liệu và dung lượng đĩa so với cách phân vùng truyền thống.

Các tệp, thuật ngữ và tiện ích sau đây đã được nhắc đến:

`/`
: Hệ thống tệp gốc của Linux.

`/var`
: Vị trí tiêu chuẩn cho “dữ liệu biến” - dữ liệu có thể thu nhỏ và phát triển theo thời gian.

`/home`
: Thư mục mẹ tiêu chuẩn cho các thư mục chính của người dùng thông thường trên hệ thống.

`/boot`
: Vị trí tiêu chuẩn cho các tệp trình tải khởi động, nhân Linux và đĩa RAM ban đầu.

Phân vùng Hệ thống EFI (ESP)
: Được sử dụng bởi các hệ thống đã triển khai UEFI để lưu trữ các tệp khởi động của hệ thống.

Không gian hoán đổi
: Được sử dụng để hoán đổi các trang bộ nhớ hạt nhân khi RAM được sử dụng nhiều.

Điểm gắn kết
: Vị trí thư mục nơi thiết bị (chẳng hạn như một ổ đĩa cứng) sẽ được gắn vào.

Phân vùng
: Sự phân chia trên một ổ đĩa cứng.

Đáp án Bài tập Hướng dẫn

1. Trên các hệ thống Linux, các tệp cho trình tải khởi động GRUB được lưu trữ ở đâu?

   Trong `/boot/grub`.

2. Phân vùng khởi động nên kết thúc ở đâu để đảm bảo rằng PC luôn có thể tải được hạt nhân?

   Trước xi lanh 1024.

3. Phân vùng EFI thường được gắn ở đâu?

   Trong `/boot/efi`.

4. Khi gắn thủ công một hệ thống tệp, nó thường được gắn vào thư mục nào?

   Trong `/mnt`. Tuy nhiên, điều này là không bắt buộc. Bạn có thể gắn kết một phân vùng trong bất kỳ thư mục nào mình muốn.

Đáp án Bài tập Mở rộng

1. Đơn vị nhỏ nhất bên trong một Nhóm Ổ phân vùng là gì?

   Nhóm Ổ phân vùng được chia nhỏ thành các mức phạm vi.

2. Kích thước của một Ổ phân vùng Logic được xác định như thế nào?

   Bằng kích thước của các mức phạm vi vật lý nhân với số lượng mức phạm vi trên ổ phân vùng.

3. Trên một đĩa được định dạng bằng sơ đồ phân vùng MBR, ID của Phân vùng hệ thống EFI là gì?

   ID là `0xEF`.

4. Bên cạnh các phân vùng hoán đổi, làm thế nào để có thể nhanh chóng tăng dung lượng hoán đổi trên hệ thống Linux?

   Có thể sử dụng tệp hoán đổi.

102.2 Cài đặt Trình Quản lý Khởi động

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 102.2

Khối lượng

2

Các lĩnh vực kiến thức chính

- Cung cấp các vị trí khởi động thay thế và các tùy chọn khởi động dự phòng.
- Cài đặt và định cấu hình bộ tải khởi động như GRUB Legacy.
- Thực hiện các thay đổi cấu hình cơ bản cho GRUB 2.
- Tương tác với trình tải khởi động.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

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

102.2 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	102 Cài đặt Linux và Quản lý Gói
Mục tiêu:	102.2 Cài đặt Trình Quản lý Khởi động
Bài:	1 trên 1

Giới thiệu

Khi máy tính được bật nguồn, phần mềm đầu tiên chạy sẽ là trình tải khởi động. Đây là một đoạn mã có mục đích duy nhất là tải hạt nhân hệ điều hành và trao quyền kiểm soát cho hạt nhân đó. Hạt nhân sẽ tải các trình điều khiển cần thiết, khởi tạo phần cứng và sau đó là tải các phần còn lại của hệ điều hành.

GRUB là trình tải khởi động được sử dụng trên hầu hết các bản phân phối Linux. Nó có thể tải nhân Linux hoặc các hệ điều hành khác (ví dụ như Windows) và có thể xử lý nhiều hình ảnh và tham số hạt nhân dưới dạng các mục menu riêng biệt. Việc lựa chọn hạt nhân khi khởi động được thực hiện thông qua giao diện điều khiển bằng bàn phím. Việc chỉnh sửa các tham số và tùy chọn khởi động sẽ được thực hiện trên giao diện dòng lệnh.

Hầu hết các bản phân phối Linux đều tự động cài đặt và định cấu hình GRUB (thực chất là GRUB 2), vì thế mà người dùng thông thường sẽ không cần phải suy nghĩ gì về việc này. Tuy nhiên, với tư cách là quản trị viên hệ thống, việc biết cách kiểm soát quá trình khởi động để có thể khôi phục hệ thống khi gặp phải lỗi khởi động (chẳng hạn như sau khi nâng cấp hạt nhân không thành công) là rất quan trọng.

Trong bài học này, chúng ta sẽ tìm hiểu về cách cài đặt, định cấu hình và tương tác với GRUB.

GRUB Legacy và GRUB 2

Phiên bản gốc của GRUB (Bộ Tải Khởi động Hợp nhất Lớn), nay được gọi là *GRUB Legacy*, được phát triển vào năm 1995 như một phần của dự án GNU Hurd và sau đó được sử dụng làm trình tải khởi động mặc định của nhiều bản phân phối Linux, thay thế các lựa chọn có từ trước đó như LILO.

GRUB 2 là một bản viết lại hoàn chỉnh của GRUB với mục đích là trở nên gọn gàng hơn, an toàn hơn và mạnh mẽ hơn. GRUB 2 có nhiều ưu điểm so với GRUB Legacy, một trong số đó là tệp cấu hình linh hoạt hơn rất nhiều (có nhiều lệnh và câu lệnh điều kiện hơn, tương tự như một ngôn ngữ tệp lệnh), thiết kế dựa nhiều trên mô-đun hơn và mang tính chất bản địa hóa/quốc tế hóa hơn.

Ngoài ra, nó cũng hỗ trợ cho các chủ đề và menu khởi động đồ họa với màn hình khởi động, khả năng khởi động LiveCD ISO trực tiếp từ ổ cứng, hỗ trợ tốt hơn cho các kiến trúc không phải x86, hỗ trợ phổ biến cho UUID (giúp xác định đĩa và phân vùng dễ dàng hơn) và nhiều hơn thế nữa.

GRUB Legacy không còn được tiếp tục phát triển (bản phát hành cuối cùng là 0,97 vào năm 2005) và ngày nay, hầu hết các bản phân phối Linux lớn đều sử dụng GRUB 2 làm trình tải khởi động mặc định. Tuy nhiên, chúng ta vẫn có thể tìm thấy các hệ thống sử dụng GRUB Legacy. Do đó, chúng ta phải biết cách sử dụng nó và biết được nó khác với GRUB 2 ở những điểm nào.

Bộ tải Khởi động nằm ở đâu?

Trước đây, các ổ đĩa cứng trên các hệ thống tương thích với PC của IBM được phân vùng bằng sơ đồ phân vùng MBR được tạo vào năm 1982 cho IBM PC-DOS (MS-DOS) 2.0.

Trong sơ đồ này, ngành 512 byte đầu tiên của đĩa được gọi là *Bản Ghi Khởi động Chính* và có chứa một bảng mô tả các phân vùng trên đĩa (bảng phân vùng) và cả mã khởi động được gọi là trình tải khởi động.

Khi máy tính được bật, mã trình tải khởi động rất nhỏ này (do hạn chế về kích thước) sẽ được tải, thực thi và chuyển quyền điều khiển tới trình tải khởi động thứ cấp trên đĩa. Bộ tải này thường nằm trong khoảng trống 32 KB giữa MBR và phân vùng đầu tiên; chúng cũng sẽ lần lượt tải (các) hệ điều hành.

Trên đĩa được phân vùng MBR, mã khởi động cho GRUB sẽ được cài đặt vào MBR. Thao tác này sẽ tải và chuyển quyền kiểm soát tới hình ảnh "lõi" (`core`) được cài đặt giữa MBR và phân vùng đầu tiên. Từ thời điểm này, GRUB sẽ có khả năng tải các phần tài nguyên cần thiết còn lại (định nghĩa menu, tệp cấu hình và các mô-đun bổ sung) từ đĩa.

Tuy nhiên, MBR có giới hạn về số lượng phân vùng (ban đầu tối đa là 4 phân vùng chính, sau này tối đa là 3 phân vùng chính với 1 phân vùng mở rộng được chia nhỏ thành một số phân vùng logic) và kích thước đĩa tối đa là 2TB. Để khắc phục những hạn chế này, sơ đồ phân vùng mới được gọi là GPT (*GUID Partition Table* - Bảng phân vùng GUID) - một phần của tiêu chuẩn UEFI (*Unified Extensible Firmware Interface*) - đã được tạo.

Ổ đĩa cứng được phân vùng GPT có thể được sử dụng với máy tính có PC BIOS truyền thống hoặc máy tính có phần sụn UEFI. Trên các máy có BIOS, phần thứ hai của GRUB sẽ được lưu trữ trong phân vùng khởi động BIOS đặc biệt.

Trên các hệ thống có phần sụn UEFI, GRUB sẽ được chương trình cơ sở tải từ các tệp `grubia32.efi` (đối với hệ thống 32-Bit) hoặc `grubx64.efi` (đối với hệ thống 64-Bit) từ một phân vùng có tên là ESP (Phân vùng hệ thống *EFI*).

Phân vùng `/boot`

Trên Linux, các tệp cần thiết cho quá trình khởi động thường được lưu trữ trên phân vùng khởi động gắn trong hệ thống tệp gốc và thường được gọi là `/boot`.

Phân vùng khởi động không phải là một yếu tố cần thiết trên các hệ thống hiện tại vì các trình tải khởi động như GRUB thường có thể tự gắn hệ thống tệp gốc và tìm kiếm các tệp cần thiết bên trong thư mục `/boot`. Tuy vậy, việc phân vùng khởi động vẫn khá là hữu ích vì nó phân tách các tệp cần thiết cho quá trình khởi động với các phần còn lại của hệ thống tệp.

Phân vùng này thường là phân vùng đầu tiên trên đĩa. Điều này là do IBM PC BIOS ban đầu đã xử lý các đĩa sử dụng *xilanh*, *đầu* và *ngành* (CHS) với tối đa 1024 xilanh, 256 đầu và 63 ngành, dẫn đến kích thước đĩa tối đa là 528 MB (504 MB trong MS-DOS) . Điều này có nghĩa là mọi thứ vượt qua mốc này sẽ không thể truy cập được trừ khi ta sử dụng sơ đồ địa chỉ đĩa khác (như *Định Địa chỉ Khối Logic* - LBA).

Vì vậy, để có thể tương thích tối đa, phân vùng `/boot` thường nằm ở đầu đĩa và kết thúc trước xi lanh 1024 (528 MB) để đảm bảo rằng máy sẽ luôn tải được hạt nhân. Kích thước khuyến nghị cho phân vùng này trên một máy hiện tại là 300 MB.

Các lý do khác cho việc có một phân vùng riêng biệt dành cho `/boot` là việc mã hóa và nén vì một số phương pháp có thể chưa được GRUB 2 hỗ trợ, hoặc nếu người dùng cần định dạng phân vùng gốc hệ thống (`/`) bằng một hệ thống tệp không được hỗ trợ.

Nội dung của Phân vùng Khởi động

Nội dung của phân vùng `/boot` có thể thay đổi theo kiến trúc hệ thống hoặc trình tải khởi động

đang được sử dụng. Trên hệ thống dựa trên x86, ta thường sẽ tìm thấy các tệp như bên dưới. Hầu hết trong số này được đặt tên với hậu tố `-VERSION`, trong đó, `-VERSION` là phiên bản của nhân Linux tương ứng. Vì vậy, nếu có một tệp cấu hình dành cho phiên bản nhân Linux `4.15.0-65-generic` thì nó sẽ được gọi là `config-4.15.0-65-generic`.

Config file

Tệp này thường được gọi là `config-VERSION` (xem ví dụ ở trên) và nó lưu trữ các tham số cấu hình cho nhân Linux. Tệp này sẽ được tạo tự động khi hạt nhân mới được biên dịch hoặc cài đặt và người dùng *không nên* trực tiếp sửa đổi nó.

System map

Tệp này là một bảng tra cứu khớp các tên biểu tượng (như biến hoặc hàm) với vị trí tương ứng của chúng trong bộ nhớ. Điều này rất hữu ích trong việc gỡ một loại lỗi hệ thống được gọi là *lỗi loạn hạt nhân* vì nó cho phép người dùng biết biến hoặc chức năng nào đã được gọi khi xảy ra lỗi. Giống như tệp cấu hình, tên của nó thường sẽ là `System.map-VERSION` (ví dụ: `System.map-4.15.0-65-generic`).

Linux kernel

Đây là hạt nhân hệ điều hành thích hợp. Tên của nó thường sẽ là `vmlinux-VERSION` (ví dụ: `vmlinux-4.15.0-65-generic`). Ta cũng có thể tìm thấy tên `vmlinuz` thay vì `vmlinux`, ký tự `z` ở cuối có nghĩa là tệp đã được nén.

Initial RAM disk

Phần này thường được gọi là `initrd.img-VERSION` và nó sẽ chứa một hệ thống tệp gốc tối thiểu được tải vào đĩa RAM. Hệ thống này sẽ chứa các tiện ích và mô-đun nhân cần thiết để hạt nhân có thể gắn kết hệ thống tệp gốc thực.

Boot loader related files

Trên các hệ thống đã được cài đặt GRUB, chúng thường sẽ nằm trong `/boot/grub` và bao gồm tệp cấu hình GRUB (`/boot/grub/grub.cfg` đối với GRUB 2 hoặc `/boot/grub/menu.lst` đối với GRUB Legacy), các mô-đun (trong `/boot/grub/i386-pc`), các tệp dịch (trong `/boot/grub/locale`) và phông chữ (trong `/boot/grub/fonts`).

GRUB 2

Cài đặt GRUB 2

GRUB 2 có thể được cài đặt bằng tiện ích `grub-install`. Nếu hệ thống không khởi động được, ta sẽ cần khởi động bằng Live CD hoặc đĩa cứu hộ, tìm phân vùng khởi động cho hệ thống, gắn nó vào rồi chạy tiện ích.

NOTE Các lệnh bên dưới sẽ giả định rằng ta đã đăng nhập với quyền gốc. Nếu không, trước tiên hãy chạy `Sudo su -` để "trở thành" siêu người dùng. Khi hoàn tất, hãy nhập `exit` để đăng xuất và trở lại thành người dùng thông thường.

Ổ đĩa cứng đầu tiên trên hệ thống thường là *thiết bị khởi động* và ta cần biết liệu có *phân vùng khởi động* có trên đĩa hay không. Điều này có thể được thực hiện với tiện ích `fdisk`. Để liệt kê tất cả các phân vùng trên đĩa đầu tiên của máy, hãy sử dụng:

```
# 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
```

Phân vùng khởi động được xác định bằng `*` bên dưới cột khởi động. Trong ví dụ trên thì nó là `/dev/sda1`.

Bây giờ, hãy tạo một thư mục tạm thời trong `/mnt` và gắn kết phân vùng bên dưới nó:

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

Sau đó, hãy chạy `grub-install`, trỏ nó tới *thiết bị* khởi động (*không* phải phân vùng) và thư mục nơi phân vùng khởi động được gắn kết. Nếu hệ thống có phân vùng khởi động chuyên dụng, lệnh sẽ là:

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

Nếu đang cài đặt vào hệ thống không có phân vùng khởi động mà chỉ có thư mục `/boot` trên hệ thống tệp gốc, hãy trỏ `grub-install` vào đó. Vì thế, lệnh sẽ là:

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

Định Cấu hình GRUB 2

Tệp cấu hình mặc định cho GRUB 2 là `/boot/grub/grub.cfg`. Tệp này được tạo tự động và không nên được chỉnh sửa thủ công. Để thay đổi cấu hình GRUB, ta cần chỉnh sửa tệp `/etc/default/grub` và sau đó là chạy tiện ích `update-grub` để tạo tệp tuân thủ.

NOTE `update-grub` thường là lối tắt đến `grub-mkconfig -o /boot/grub/grub.cfg`, vì thế nên chúng sẽ tạo ra các kết quả giống nhau.

Có một số tùy chọn trong tệp `/etc/default/grub` sẽ kiểm soát hành vi của GRUB 2 như hạt nhân mặc định để khởi động, thời gian chờ, tham số dòng lệnh bổ sung, v.v. Những tùy chọn quan trọng nhất là:

`GRUB_DEFAULT=`

Mục menu mặc định để khởi động. Đây có thể là một giá trị số (như `0`, `1`, v.v.), tên của một mục menu (như `debian`) hoặc `saved` được sử dụng cùng với `GRUB_SAVEDEFAULT=` và sẽ được giải thích ở bên dưới . Hãy nhớ rằng các mục menu sẽ bắt đầu từ 0; vì vậy, mục menu đầu tiên sẽ là `0`, mục thứ hai sẽ là `1`, v.v.

`GRUB_SAVEDEFAULT=`

Nếu tùy chọn này được đặt thành `true` và `GRUB_DEFAULT=` được đặt thành `saved` thì tùy chọn khởi động mặc định sẽ luôn là tùy chọn cuối cùng được chọn trong menu khởi động.

`GRUB_TIMEOUT=`

Thời gian chờ được tính bằng giây trước khi mục menu mặc định được chọn. Nếu được đặt thành `0`, hệ thống sẽ khởi động mục nhập mặc định mà không hiển thị menu. Nếu được đặt thành `-1`, hệ thống sẽ đợi cho tới khi người dùng chọn một tùy chọn, bất kể là mất bao lâu.

`GRUB_CMDLINE_LINUX=`

Tuỳ chọn sẽ này liệt kê các tùy chọn dòng lệnh được thêm vào các mục cho nhân Linux.

`GRUB_CMDLINE_LINUX_DEFAULT=`

Theo mặc định, hai mục menu sẽ được tạo cho mỗi nhân Linux, một mục có các tùy chọn mặc định và một mục dùng để khôi phục. Với tùy chọn này, ta có thể thêm các tham số bổ sung và chúng sẽ chỉ được thêm vào mục nhập mặc định.

`GRUB_ENABLE_CRYPTODISK=`

Nếu được đặt thành `y`, các lệnh như `grub-mkconfig`, `update-grub` và `grub-install` sẽ tìm

các đĩa được mã hóa và thêm các lệnh cần thiết để truy cập chúng trong khi khởi động. Việc này sẽ vô hiệu hóa khả năng khởi động tự động (`GRUB_TIMEOUT=` với bất kỳ giá trị nào khác `-1`) vì sẽ cần có cụm mật khẩu để giải mã các đĩa trước khi có thể truy cập chúng.

Quản lý Mục Menu

Khi `update-grub` được chạy, GRUB 2 sẽ quét các hạt nhân và hệ điều hành trên máy và tạo các mục menu tương ứng trên tệp `/boot/grub/grub.cfg`. Các mục mới có thể được thêm thủ công vào các tệp tệp lệnh trong thư mục `/etc/grub.d`.

Các tệp này phải có thể thực thi được và được `update-grub` xử lý theo thứ tự số. Do đó, `05_debian_theme` sẽ được xử lý trước `10_linux`, v.v. Các mục menu tùy chỉnh thường được thêm vào tệp `40_custom`.

Cú pháp cơ bản cho một mục menu sẽ giống như sau:

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

Dòng đầu tiên sẽ luôn bắt đầu bằng `menuentry` và kết thúc bằng `{`. Văn bản giữa các dấu ngoặc kép sẽ được hiển thị dưới dạng nhãn của mục trên menu khởi động GRUB 2.

Tham số `set root` sẽ xác định đĩa và phân vùng chứa hệ thống tệp gốc cho hệ điều hành. Hãy lưu ý rằng trên GRUB, 2 đĩa sẽ được đánh số từ 0; vì vậy, `hd0` sẽ là đĩa đầu tiên (`sda` trong Linux), `hd1` là đĩa thứ hai, v.v. Tuy nhiên, các phân vùng lại được đánh số bắt đầu từ một. Trong ví dụ trên, hệ thống tệp gốc nằm ở đĩa đầu tiên (`hd0`), phân vùng đầu tiên (`,1`) hoặc `sda1`.

Thay vì chỉ định trực tiếp thiết bị và phân vùng, ta cũng có thể yêu cầu GRUB 2 tìm kiếm hệ thống tệp có nhãn cụ thể hoặc UUID (*Số Nhận dạng Duy nhất Toàn cầu*). Đối với việc này, hãy sử dụng tham số `search --set=root`, theo sau là tham số `--label` và nhãn hệ thống tệp để tìm kiếm, hoặc `--fs-uuid` theo sau là UUID của hệ thống tệp.

Ta có thể tìm UUID của hệ thống tệp bằng lệnh dưới đây:

```
$ 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
```

Trong ví dụ trên, UUID cho `/dev/sda1` là `ae71b214-0aec-48e8-80b2-090b6986b625`. Nếu bạn muốn đặt nó làm thiết bị gốc cho GRUB 2, lệnh sẽ là `search --set=root --fs-uuid ae71b214-0aec-48e8-80b2-090b6986b625`.

Khi sử dụng lệnh `search`, người ta thường thêm tham số `--no-floppy` để GRUB không mất thời gian tìm kiếm trên đĩa mềm.

Dòng `linux` cho biết vị trí đặt hạt nhân của hệ điều hành (trong trường hợp này là tệp `vmlinuz` ở thư mục gốc của hệ thống tệp). Sau đó, ta có thể truyền tham số dòng lệnh đến hạt nhân.

Trong ví dụ trên, chúng ta đã chỉ định phân vùng gốc (`root=/dev/sda1`) và chuyển ba tham số hạt nhân: phân vùng gốc phải được gắn ở chế độ chỉ đọc (`ro`), hầu hết các thông báo tường trình sẽ bị vô hiệu hóa (` quiet `) và một màn hình giật gân sẽ được hiển thị (`splash`).

Dòng `initrd` cho biết vị trí của đĩa RAM ban đầu. Trong ví dụ trên, tệp là `initrd.img` nằm ở thư mục gốc của hệ thống tệp.

NOTE Hầu hết các bản phân phối Linux không thực sự đặt hạt nhân và initrd vào thư mục gốc của hệ thống tệp gốc. Thay vào đó, đây là các liên kết đến các tệp thực tế bên trong thư mục hoặc phân vùng `/boot`.

Dòng cuối cùng của mục menu chỉ được chứa ký tự `}`.

Tương tác với GRUB 2

Khi khởi động hệ thống bằng GRUB 2, ta sẽ thấy một menu các tùy chọn. Hãy sử dụng các phím mũi tên để chọn một tùy chọn và phím `Enter` để xác nhận và khởi động mục đã chọn.

TIP Nếu bạn chỉ thấy đồng hồ đếm ngược chứ không thấy menu, hãy nhấn phím `Shift` để hiển thị menu.

Để chỉnh sửa một tùy chọn, hãy chọn tùy chọn đó bằng các phím mũi tên và nhấn phím `E` để hiển thị một cửa sổ soạn thảo với nội dung của `menuentry` được liên kết với tùy chọn đó như đã được xác định trong `/boot/grub/grub.cfg`.

Sau khi chỉnh sửa một tùy chọn, hãy nhập `Ctrl` + `X` hoặc `F10` để khởi động hoặc `Esc` để quay lại menu.

Để vào vỏ GRUB 2, hãy nhấn `C` trên màn hình menu (hoặc `Ctrl` + `C`) trên cửa sổ soạn thảo) và ta sẽ thấy một dấu nhắc lệnh như thế này: `grub >`

Hãy nhập `help` để xem danh sách tất cả các lệnh có sẵn hoặc nhấn `Esc` để thoát khỏi vỏ và quay lại màn hình menu.

NOTE Hãy nhớ rằng menu này sẽ không xuất hiện nếu `GRUB_TIMEOUT` được đặt thành `0` trong `/etc/default/grub`.

Khởi động từ Vỏ GRUB 2

Ta có thể sử dụng vỏ GRUB 2 để khởi động hệ thống trong trường hợp cấu hình sai trong mục nhập menu khiến nó bị lỗi.

Điều đầu tiên chúng ta nên làm là tìm xem phân vùng khởi động ở đâu. Ta có thể làm điều đó bằng lệnh `ls`; lệnh này sẽ hiển thị danh sách các phân vùng và đĩa mà GRUB 2 đã tìm thấy.

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

Trong ví dụ trên, mọi thứ rất dễ dàng vì chỉ có một đĩa `(hd0)` với duy nhất một phân vùng trên đó là `(hd0,msdos1)`.

Các đĩa và phân vùng nào được liệt kê sẽ khác nhau tuỳ thuộc từng hệ thống. Trong ví dụ của chúng ta, phân vùng đầu tiên của `hd0` được gọi là `msdos1` vì đĩa được phân vùng bằng sơ đồ phân vùng MBR. Nếu nó được phân vùng bằng GPT, tên sẽ là `gpt1`.

Để khởi động Linux, chúng ta cần một hạt nhân và đĩa RAM ban đầu (initrd). Hãy cùng kiểm tra nội dung của `(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/
```

Chúng ta có thể thêm tham số `-l` vào `ls` để có một danh sách dài tương tự như những gì ta sẽ nhận được trên cửa sổ dòng lệnh Linux. Hãy sử dụng `Tab` để tự động điền tên đĩa, phân vùng và tên tệp.

Hãy lưu ý rằng chúng ta có một hình ảnh hạt nhân (`vmlinuz`) và initrd (`initrd.img`) ngay trên thư mục gốc. Nếu không, chúng ta có thể kiểm tra nội dung của `/boot` với `list (hd0,msdos1)/boot/`.

Bây giờ, hãy đặt phân vùng khởi động:

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

Hãy tải nhân Linux bằng lệnh `linux`, theo sau là đường dẫn đến hạt nhân và tùy chọn `root=` để báo cho hạt nhân biết vị trí của hệ thống tệp gốc cho hệ điều hành.

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

Tải đĩa RAM ban đầu bằng `initrd`, theo sau là đường dẫn đầy đủ đến tệp `initrd.img`:

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

Bây giờ, hãy khởi động hệ thống với `boot`.

Khởi động từ Vỏ Cứu hộ

Trong trường hợp khởi động không thành công, GRUB 2 có thể tải vỏ cứu hộ (rescue shell) - một phiên bản đơn giản hóa của vỏ mà chúng ta đã đề cập tới trước đây. Chúng ta sẽ nhận ra nó bằng dấu nhắc lệnh hiển thị `grub rescue>`.

Quá trình khởi động hệ thống từ vỏ này cũng sẽ gần giống như đã được trình bày ở trên. Tuy nhiên, chúng ta sẽ cần tải một vài mô-đun GRUB 2 để hệ thống có thể hoạt động.

Sau khi tìm ra phân vùng nào là phân vùng khởi động (với `ls`), hãy sử dụng lệnh `set prefix=`, theo sau là đường dẫn đầy đủ đến thư mục chứa các tệp GRUB 2, thường sẽ là `/boot/grub`. Trong ví dụ của chúng ta:

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

Bây giờ, hãy tải các mô-đun `normal` và `linux` bằng lệnh `insmod`:

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

Tiếp theo, hãy đặt phân vùng khởi động bằng `set root=` như hướng dẫn ở trên, tải nhân linux (với `linux`), đĩa RAM ban đầu (`initrd`) và thử khởi động bằng `boot`.

GRUB Legacy

Cài đặt GRUB Legacy từ Hệ thống đang chạy

Để cài đặt GRUB Legacy trên đĩa từ hệ thống đang chạy, chúng ta có thể sử dụng tiện ích `grub-install`. Lệnh cơ bản là `grub-install DEVICE`, trong đó, `DEVICE` là đĩa cần cài đặt GRUB Legacy ở đó. Một ví dụ có thể là `/dev/sda`.

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

Hãy lưu ý rằng ta cần chỉ định *thiết bị* nơi GRUB Legacy sẽ được cài đặt (ví dụ như `/dev/sda/`) chứ *không phải phân vùng* như trong `/dev/sda1`.

Theo mặc định, GRUB sẽ sao chép các tệp cần thiết vào thư mục `/boot` trên thiết bị được chỉ định. Nếu muốn sao chép chúng sang một thư mục khác, ta có thể sử dụng tham số `--boot-directory=`, theo sau là đường dẫn đầy đủ đến nơi các tệp sẽ được sao chép vào.

Cài đặt GRUB Legacy từ Vỏ GRUB

Nếu không thể khởi động hệ thống vì một lý do nào đó và cần cài đặt lại GRUB Legacy, chúng ta có thể thực hiện điều này từ Vỏ GRUB trên đĩa khởi động GRUB Legacy.

Từ vỏ GRUB (nhập `c` tại menu khởi động để đến dấu nhắc `grub>`), bước đầu tiên là đặt thiết bị khởi động chứa thư mục `/boot`. Ví dụ: nếu thư mục này nằm trong phân vùng đầu tiên của đĩa đầu tiên, lệnh sẽ là:

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

Nếu không biết thiết bị nào chứa thư mục `/boot`, ta có thể yêu cầu GRUB tìm kiếm nó bằng lệnh `find` như dưới đây:

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

Sau đó, hãy thiết lập phân vùng khởi động như hướng dẫn ở trên, sử dụng lệnh `setup` để cài đặt GRUB Legacy vào MBR và sao chép các tệp cần thiết vào đĩa:

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

Khi hoàn tất khởi động lại hệ thống, nó sẽ khởi động một cách bình thường.

Định cấu hình các Mục Nhập và Cài đặt Menu của GRUB Legacy

Các mục nhập và cài đặt menu của GRUB Legacy được lưu trữ trong tệp `/boot/grub/menu.lst`. Đây là một tệp văn bản đơn giản với danh sách các lệnh và tham số, có thể được chỉnh sửa trực tiếp bằng trình soạn thảo văn bản mà bạn yêu thích.

Các dòng bắt đầu bằng `#` được coi là chú thích và các dòng trống sẽ bị bỏ qua.

Một mục menu có ít nhất ba lệnh. Lệnh đầu tiên là `title`, dùng để đặt tiêu đề của hệ điều hành trên màn hình menu. Lệnh thứ hai là `root` sẽ cho GRUB Legacy biết khởi động từ thiết bị hoặc phân vùng nào.

Lệnh thứ ba là `kernel` và được dùng để chỉ định đường dẫn đầy đủ đến hình ảnh hạt nhân sẽ được tải khi mục tương ứng được chọn. Hãy lưu ý rằng đường dẫn này có liên quan đến thiết bị được chỉ định trên tham số `root`.

Một ví dụ đơn giản như sau:

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

Không giống như GRUB 2, trong GRUB Legacy, cả phân vùng *và* đĩa đều được đánh số từ 0. Vì vậy, lệnh `root (hd0,0)` sẽ đặt phân vùng khởi động là phân vùng đầu tiên (`0`) của đĩa đầu tiên (`hd0`).

Chúng ta có thể bỏ qua câu lệnh `root` khi chỉ định thiết bị khởi động trước đường dẫn trên lệnh `kernel`. Cú pháp của chúng giống nhau; vì vậy:

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

tương đương với:

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

Cả hai đều sẽ tải tệp `vmlinuz` từ thư mục gốc (`/`) của phân vùng đầu tiên của đĩa đầu tiên (`hd0,0`).

Tham số `root=/dev/hda1` sau lệnh `kernel` sẽ cho nhân Linux biết phân vùng nào sẽ được sử dụng làm hệ thống tệp gốc. Đây là tham số nhân Linux, không phải một lệnh GRUB Legacy.

NOTE Để tìm hiểu thêm về các tham số hạt nhân, hãy xem https://www.kernel.org/doc/html/v4.14/admin-guide/kernel-parameters.html.

Chúng ta có thể sẽ cần chỉ định vị trí hình ảnh của Đĩa RAM ban đầu cho hệ điều hành với tham số `initrd`. Đường dẫn đầy đủ đến tệp có thể được chỉ định như trong tham số `kernel` và ta cũng có thể chỉ định thiết bị hoặc phân vùng trước đường dẫn, ví dụ:

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

GRUB Legacy có thiết kế dựa trên mô-đun; trong đó, các mô-đun (thường được lưu trữ dưới dạng tệp `.mod` trong `/boot/grub/i386-pc`) có thể được tải để thêm các tính năng bổ sung như các hỗ trợ cho phần cứng bất thường, hệ thống tệp hoặc thuật toán nén mới .

Các mô-đun có thể được tải bằng cách sử dụng lệnh `module`, theo sau là đường dẫn đầy đủ đến tệp `.mod` tương ứng. Hãy nhớ rằng, giống như hạt nhân và hình ảnh initrd, đường dẫn này có liên quan đến thiết bị được chỉ định trong lệnh `root`.

Ví dụ bên dưới sẽ tải mô-đun `915resolution` cần thiết cho việc thiết lập chính xác độ phân giải của bộ nhớ đệm khung trên các hệ thống có bộ vi xử lý video chuỗi sê-ri 800 hoặc 900.

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

Tải Chuỗi Hệ điều hành khác

GRUB Legacy có thể được sử dụng để tải các hệ điều hành không được hỗ trợ (chẳng hạn như Windows) sử dụng tiến trình có tên là *tải chuỗi* (chainloading). GRUB Legacy sẽ được tải trước và khi tùy chọn tương ứng được chọn, trình tải khởi động cho hệ thống mong muốn sẽ được tải.

Một mục điển hình cho việc tải chuỗi Windows sẽ giống như dưới đây:

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

```
chainload +1
boot
```

Hãy cùng xem từng tham số. Như trước đây, `root (hd0,1)` sẽ chỉ định thiết bị và phân vùng nơi đặt trình tải khởi động cho hệ điều hành cần tải. Trong ví dụ này sẽ là phân vùng *thứ hai* của đĩa đầu tiên.

`makeactive`

sẽ đặt cờ cho biết đây là phân vùng đang hoạt động. Điều này chỉ đúng trên các phân vùng chính của DOS.

`chainload +1`

yêu cầu GRUB tải ngành đầu tiên của phân vùng khởi động. Đây là nơi thường đặt trình tải khởi động.

`boot`

sẽ chạy trình tải khởi động và tải hệ điều hành tương ứng.

Bài tập Hướng dẫn

1. Vị trí mặc định cho tệp cấu hình GRUB 2 là gì?

2. Các bước cần thiết để thay đổi cài đặt cho GRUB 2 là gì?

3. Nên thêm các mục menu GRUB 2 tùy chỉnh vào tệp nào?

4. Các mục menu cho GRUB Legacy được lưu trữ ở đâu?

5. Từ menu GRUB 2 hoặc GRUB Legacy, làm cách nào để có thể vào Vỏ GRUB?

Bài tập Mở rộng

1. Hãy tưởng tượng một người dùng đang định cấu hình GRUB Legacy để khởi động từ phân vùng thứ hai của đĩa đầu tiên. Họ đã viết mục menu tùy chỉnh sau:

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

   Tuy nhiên, hệ thống lại không khởi động. Hãy giải thích tại sao.

2. Hãy tưởng tượng bạn có một đĩa được xác định là `/dev/sda` với nhiều phân vùng. Lệnh nào có thể được sử dụng để tìm ra đâu là phân vùng khởi động trên hệ thống?

3. Lệnh nào có thể được sử dụng để tìm ra UUID của một phân vùng?

4. Hãy xem xét mục sau cho GRUB 2.

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

   Hãy thay đổi nó để hệ thống khởi động từ đĩa với UUID `5dda0af3-c995-481a-a6f3-46dcd3b6998d`

5. Làm cách nào để có thể đặt GRUB 2 đợi trong 10 giây trước khi khởi động mục menu mặc định?

6. Từ Vỏ GRUB Legacy, các lệnh để cài đặt GRUB vào phân vùng đầu tiên của đĩa thứ hai là gì?

Tóm tắt

Trong bài học này, chúng ta đã học về:

- Trình tải khởi động là gì.
- Sự khác biệt giữa GRUB Legacy và GRUB 2.
- Phân vùng khởi động và nội dung của nó là gì.
- Cách cài đặt GRUB Legacy và GRUB 2.
- Cách định cấu hình GRUB Legacy và GRUB 2.
- Cách thêm các mục menu tùy chỉnh vào GRUB Legacy và GRUB 2.
- Cách tương tác với màn hình menu và bảng điều khiển của GRUB Legacy và GRUB 2.
- Cách khởi động hệ thống từ GRUB Legacy hoặc vỏ GRUB 2 hoặc vỏ cứu hộ.

Các lệnh sau đã được thảo luận trong bài học này:

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

Đáp án Bài tập Hướng dẫn

1. Vị trí mặc định cho tệp cấu hình GRUB 2 là gì?

   `/boot/grub/grub.cfg`

2. Các bước cần thiết để thay đổi cài đặt cho GRUB 2 là gì?

   Hãy thực hiện các thay đổi đối với tệp `/etc/default/grub`, sau đó cập nhật cấu hình bằng `update-grub`.

3. Nên thêm các mục menu GRUB 2 tùy chỉnh vào tệp nào?

   `/etc/grub.d/40_custom`

4. Các mục menu cho GRUB Legacy được lưu trữ ở đâu?

   `/boot/grub/menu.lst`

5. Từ menu GRUB 2 hoặc GRUB Legacy, làm cách nào để có thể vào Vỏ GRUB?

   Nhấn `c` trong màn hình menu.

Đáp án Bài tập Mở rộng

1. Hãy tưởng tượng một người dùng đang định cấu hình GRUB Legacy để khởi động từ phân vùng thứ hai của đĩa đầu tiên. Họ đã viết mục menu tùy chỉnh sau:

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

   Tuy nhiên, hệ thống lại không khởi động. Hãy giải thích tại sao.

   Phân vùng khởi động đã bị sai. Hãy nhớ rằng, không giống như GRUB 2, GRUB Legacy sẽ đếm các phân vùng từ *số không*. Vì vậy, lệnh đúng cho phân vùng thứ hai của đĩa đầu tiên phải là `root (hd0,1)`.

2. Hãy tưởng tượng bạn có một đĩa được xác định là `/dev/sda` với nhiều phân vùng. Lệnh nào có thể được sử dụng để tìm ra đâu là phân vùng khởi động trên hệ thống?

   Sử dụng `fdisk -l /dev/sda`. Phân vùng khởi động sẽ được đánh dấu hoa thị (`*`) trong danh sách.

3. Lệnh nào có thể được sử dụng để tìm ra UUID của một phân vùng?

   Sử dụng `ls -la /dev/disk/by-uuid/` và tìm UUID trỏ đến phân vùng.

4. Hãy xem xét mục sau cho GRUB 2.

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

   Hãy thay đổi nó để hệ thống khởi động từ đĩa với UUID `5dda0af3-c995-481a-a6f3-46dcd3b6998d`

   Ta sẽ cần thay đổi câu lệnh `set root`. Thay vì chỉ định đĩa và phân vùng, hãy yêu cầu grub tìm kiếm phân vùng có UUID mong muốn.

   ```
   menuentry "Default OS" {
   ```

```
  search --set=root --fs-uuid 5dda0af3-c995-481a-a6f3-46dcd3b6998d
  linux /vmlinuz root=/dev/sda1 ro quiet splash
  initrd /initrd.img
}
```

5. Làm cách nào để có thể đặt GRUB 2 đợi trong 10 giây trước khi khởi động mục menu mặc định?

   Thêm tham số `GRUB_TIMEOUT=10` vào `/etc/default/grub`.

6. Từ Vỏ GRUB Legacy, các lệnh để cài đặt GRUB vào phân vùng đầu tiên của đĩa thứ hai là gì?

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

102.3 Quản lý Thư viện chia sẻ

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 102.3

Khối lượng

1

Các lĩnh vực kiến thức chính

- Xác định các thư viện chia sẻ.
- Xác định các vị trí điển hình của thư viện hệ thống.
- Tải thư viện chia sẻ.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

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

102.3 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	102 Cài đặt Linux và Quản lý Gói
Mục tiêu:	102.3 Cài đặt Trình Quản lý Khởi động
Bài:	1 trên 1

Giới thiệu

Trong bài học này, chúng ta sẽ thảo luận về *thư viện chia sẻ* hay còn được gọi là *đối tượng chia sẻ*: các đoạn mã được biên dịch, có thể tái sử dụng như các hàm hoặc hạng và được nhiều chương trình khác nhau thường xuyên sử dụng.

Để bắt đầu, chúng ta sẽ tìm hiểu thư viện chia sẻ là gì, cách xác định chúng và nơi ta có thể tìm thấy chúng. Tiếp theo, chúng ta sẽ đi vào cách định cấu hình các vị trí lưu trữ của chúng. Cuối cùng, chúng ta sẽ tìm hiểu về cách tìm kiếm các thư viện chia sẻ mà một chương trình cụ thể sẽ phụ thuộc vào.

Khái niệm về Thư viện Chia sẻ

Tương tự như phiên bản vật lý của chúng, thư viện phần mềm là bộ sưu tập mã nhằm phục vụ cho nhiều chương trình khác nhau, cũng giống như các thư viện vật lý giữ sách và các tài nguyên để nhiều người có thể sử dụng.

Để tạo một tệp thực thi từ mã nguồn của chương trình, ta cần có hai bước quan trọng. Đầu tiên, *trình biên dịch* sẽ biến mã nguồn thành mã máy được lưu trữ trong cái gọi là *tệp đối tượng*. Thứ

hai, *trình liên kết* sẽ kết hợp các tệp đối tượng và *liên kết* chúng với các thư viện để tạo ra tệp thực thi cuối cùng. Liên kết này có thể được thực hiện theo phương thức *tĩnh* hoặc *động*. Tùy thuộc vào phương thức được sử dụng, chúng ta sẽ thảo luận về thư viện tĩnh hoặc về thư viện chia sẻ trong trường hợp liên kết động. Hãy cùng giải thích sự khác biệt giữa chúng.

Thư viện tĩnh

Một thư viện tĩnh được hợp nhất với chương trình tại thời điểm liên kết. Một bản sao của mã thư viện sẽ được nhúng vào chương trình và trở thành một phần của nó. Do đó, chương trình sẽ không phụ thuộc vào thư viện trong thời gian chạy vì chương trình đã có chứa mã thư viện. Không có phần phụ thuộc có thể được coi là một lợi thế vì chúng ta sẽ không phải lo lắng về việc đảm bảo các thư viện được sử dụng luôn luôn sẵn sàng. Mặt khác, các chương trình được liên kết tĩnh thường sẽ nặng hơn.

Thư viện chia sẻ (hoặc động)

Trong trường hợp của thư viện chia sẻ, trình liên kết chỉ cần đảm bảo rằng chương trình tham chiếu được chính xác các thư viện. Tuy nhiên, trình liên kết sẽ không sao chép bất kỳ mã thư viện nào vào tệp chương trình. Dù vậy, trong thời gian chạy, thư viện chia sẻ sẽ phải có sẵn để đáp ứng các phần phụ thuộc của chương trình. Đây là một cách tiếp cận mang tính kinh tế để quản lý tài nguyên hệ thống vì nó sẽ giúp giảm kích thước tệp chương trình và chỉ duy nhất một bản sao của thư viện được tải vào bộ nhớ ngay cả khi nó được sử dụng bởi nhiều chương trình.

Quy ước đặt tên Tệp Đối tượng Chia sẻ

Tên của một thư viện chia sẻ (hay còn được gọi là *soname*) được đặt tuân theo một mẫu tạo bởi ba thành phần:

- Tên thư viện (thường có tiền tố là `lib` trong library - tức thư viện)
- `so` (viết tắt của “shared object” - tức đối tượng chia sẻ)
- Số phiên bản của thư viện

Ví dụ: `libpthread.so.0`

Ngược lại, tên của thư viện tĩnh sẽ kết thúc bằng `.a`, ví dụ như `libpthread.a`.

NOTE

Bởi vì các tệp chứa thư viện chia sẻ phải có sẵn khi chương trình được thực thi, hầu hết các hệ thống Linux đều có chứa thư viện chia sẻ. Vì các thư viện tĩnh chỉ được yêu cầu trong một tệp chuyên dụng khi một chương trình được liên kết nên chúng có thể sẽ không có trên hệ thống người dùng cuối.

`glibc` (thư viện GNU C) là một ví dụ điển hình về thư viện chia sẻ. Trên hệ thống Debian GNU/Linux 9.9, tệp của nó được đặt tên là `libc.so.6`. Các tên tệp chung chung như vậy thường là các liên kết tượng trưng trỏ đến tệp thực chứa một thư viện, tên của tệp sẽ chứa số phiên bản chính xác. Trong trường hợp của `glibc`, liên kết tượng trưng này sẽ giống như sau:

```
$ 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
```

Việc tham chiếu các tệp thư viện chia sẻ (được đặt tên theo một phiên bản cụ thể) tới các tên tệp chung chung hơn là khá phổ biến.

Các ví dụ khác về thư viện chia sẻ bao gồm `libreadline` (cho phép người dùng chỉnh sửa các dòng lệnh khi chúng được nhập và bao gồm hỗ trợ cho cả chế độ chỉnh sửa Emacs và vi), `libcrypt` (chứa các chức năng liên quan đến mã hóa, hàm băm và mã hóa) hoặc `libcurl` (là thư viện truyền tệp đa giao thức).

Các vị trí phổ biến cho các thư viện chia sẻ trong hệ thống Linux là:

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

NOTE Khái niệm thư viện chia sẻ không chỉ dành riêng cho Linux. Ví dụ như trong Windows, chúng được gọi là DLL, viết tắt của *dynamic linked libraries*, tức thư viện được liên kết động.

Cấu hình của Đường dẫn Thư viện Chia sẻ

Các tham chiếu trong các chương trình được liên kết động sẽ được giải quyết bởi trình liên kết động (`ld.so` hoặc `ld-linux.so`) khi chương trình được chạy. Trình liên kết động sẽ tìm kiếm các thư viện trong một số thư mục. Các thư mục này được chỉ định bởi *đường dẫn thư viện*. Đường dẫn thư viện được định cấu hình trong thư mục `/etc`, cụ thể là trong tệp `/etc/ld.so.conf` và phổ biến hơn hiện nay là trong các tệp trong thư mục `/etc/ld.so.conf.d` . Thông thường, `/etc/ld.so.conf` chỉ bao gồm một dòng đơn `include` cho các tệp `*.conf` trong `/etc/ld.so.conf.d`:

```
$ cat /etc/ld.so.conf
```

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

Thư mục `/etc/ld.so.conf.d` có chứa các tệp `*.conf`:

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

Các tệp `*.conf` này phải bao gồm các đường dẫn tuyệt đối đến các thư mục thư viện chia sẻ:

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

Lệnh `ldconfig` đảm nhiệm việc đọc các tệp cấu hình này, tạo tập hợp các liên kết tượng trưng như đã nói ở trên giúp định vị các thư viện riêng lẻ và cuối cùng là cập nhật tệp bộ nhớ đệm `/etc/ld.so.cache`. Do đó, `ldconfig` phải được chạy mỗi khi tệp cấu hình được thêm hoặc cập nhật.

Các tùy chọn hữu ích cho `ldconfig` là:

`-v`, `--verbose`

Hiển thị số phiên bản thư viện, tên của từng thư mục và các liên kết được tạo:

```
$ 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
    (...)
```

Vì vậy, chúng ta có thể thấy được cách `libfuse.so.2` được liên kết với tệp đối tượng chia sẻ thực tế `libfuse.so.2.9.7`.

`-p`, `--print-cache`

In danh sách các thư mục và thư viện ứng cử viên được lưu trữ trong bộ nhớ đệm hiện tại:

```
$ 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
    (...)
```

Hãy lưu ý cách bộ nhớ đệm sử dụng soname đủ điều kiện của các liên kết:

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

Nếu liệt kê dài `/lib/x86_64-linux-gnu/libfuse.so.2`, chúng ta sẽ tìm thấy tham chiếu đến tệp đối tượng chia sẻ thực tế `libfuse.so.2.9.7` được lưu trữ trong cùng một thư mục:

```
$ 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

Vì nó yêu cầu quyền ghi vào `/etc/ld.so.cache` (do siêu người dùng sở hữu) nên ta sẽ phải là siêu người dùng hoặc sử dụng `sudo` để gọi `ldconfig`. Để biết thêm thông tin về các khoá chuyển của `ldconfig`, hãy tham khảo trang hướng dẫn của nó.

Ngoài các tệp cấu hình được mô tả ở trên, biến môi trường `LD_LIBRARY_PATH` có thể được sử dụng để tạm thời thêm đường dẫn mới cho thư viện chia sẻ. Nó được tạo thành từ một tập hợp các thư mục được phân tách bằng dấu hai chấm (`:`) nơi các thư viện được tra cứu. Ví dụ: để thêm `/usr/local/mylib` vào đường dẫn thư viện trong phiên vỏ hiện tại, ta có thể nhập:

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

Bây giờ, ta có thể kiểm tra giá trị của nó:

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

Để thêm `/usr/local/mylib` vào đường dẫn thư viện trong phiên vỏ hiện tại và xuất nó sang tất cả các tiến trình con được sinh ra từ vỏ đó, ta nhập:

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

Để xóa biến môi trường `LD_LIBRARY_PATH`, ta chỉ cần gõ:

```
$ unset LD_LIBRARY_PATH
```

Để làm cho thay đổi trở thành vĩnh viễn, ta có thể viết:

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

vào trong một trong các tập lệnh khởi tạo của Bash, chẳng hạn như `/etc/bash.bashrc` hoặc `~/.bashrc`.

NOTE `LD_LIBRARY_PATH` đối với các thư viện chia sẻ cũng giống như `PATH` đối với các tệp thực thi được. Để biết thêm thông tin về các biến môi trường và cấu hình vỏ, hãy tham khảo các bài học tương ứng.

Tìm kiếm các Phần Phụ thuộc của một Tệp Thực thi cụ thể

Để tra cứu các thư viện chia sẻ theo yêu cầu của một chương trình cụ thể, hãy sử dụng lệnh `ldd`, theo sau là đường dẫn tuyệt đối tới chương trình. Đầu ra sẽ hiển thị đường dẫn của tệp thư viện chia sẻ cũng như địa chỉ bộ nhớ hệ thập lục phân nơi nó được tải:

```
$ ldd /usr/bin/git
    linux-vdso.so.1 =>  (0x00007ffcbb310000)
    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)
```

Tương tự như vậy, chúng ta có thể sử dụng `ldd` để tìm kiếm các phần phụ thuộc của một đối tượng chia sẻ:

```
$ ldd /lib/x86_64-linux-gnu/libc.so.6
    /lib64/ld-linux-x86-64.so.2 (0x00007fbfed578000)
    linux-vdso.so.1 (0x00007fffb7bf5000)
```

Với tùy chọn `-u` (hoặc `--unused`), `ldd` sẽ in các phần phụ thuộc trực tiếp không được sử dụng (nếu chúng tồn tại):

```
$ 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
```

Lý do cho các phần phụ thuộc không được sử dụng có liên quan đến các tùy chọn được trình liên kết sử dụng khi xây dựng tệp nhị phân. Mặc dù chương trình không cần tới một thư viện không được sử dụng, nó vẫn sẽ được liên kết và gắn nhãn là `NEEDED` trong thông tin về tệp đối tượng. Ta có thể xem kỹ điều này bằng cách sử dụng các lệnh như `readelf` hoặc `objdump` mà chúng ta sẽ sớm phải áp dụng trong phần bài tập mở rộng.

Bài tập Hướng dẫn

1. Hãy chia các tên thư viện chia sẻ sau thành các phần của chúng:

Tên Tệp đầy đủ	**Tên Thư viện**	**Hậu tố *so***	**Số phiên bản**
`linux-vdso.so.1`			
`libprocps.so.6`			
`libdl.so.2`			
`libc.so.6`			
`libsystemd.so.0`			
`ld-linux-x86-64.so.2`			

2. Bạn đã phát triển một phần mềm và muốn thêm một thư mục thư viện chia sẻ mới vào hệ thống của mình (`/opt/lib/mylib`). Bạn đã viết đường dẫn tuyệt đối của nó trong tệp có tên `mylib.conf`.
   - Bạn nên đặt tệp này trong thư mục nào?
   - Bạn nên chạy lệnh nào để các thay đổi có hiệu lực hoàn toàn?

3. Bạn sẽ sử dụng lệnh nào để liệt kê các thư viện chia sẻ theo yêu cầu của `kill`?

Bài tập Mở rộng

1. `objdump` là một tiện ích dòng lệnh hiển thị thông tin từ các tệp đối tượng. Hãy kiểm tra xem nó đã được cài đặt trong hệ thống hay chưa với `which objdump`. Nếu chưa thì bạn hãy cài đặt nó ngay bây giờ.
   - Hãy sử dụng `objdump` với tùy chọn `-p` (hoặc `--private-headers`) và `grep` để in các phần phụ thuộc của `glibc`:
   - Hãy sử dụng `objdump` với tùy chọn `-p` (hoặc `--private-headers`) và `grep` để in soname của `glibc`:
   - Hãy sử dụng `objdump` với tùy chọn `-p` (hoặc `--private-headers`) và `grep` để in các phần phụ thuộc của Bash:

Tóm tắt

Trong bài học này, chúng ta đã học về:

- Thư viện chia sẻ (hoặc thư viện động) là gì.
- Sự khác nhau giữa thư viện chia sẻ và thư viện tĩnh.
- Tên của các thư viện chia sẻ (*sonames*).
- Các vị trí thích hợp dành cho thư viện chia sẻ trong hệ thống Linux, chẳng hạn như `/lib` hoặc `/usr/lib`.
- Mục đích của trình liên kết động `ld.so` (hoặc `ld-linux.so`).
- Cách định cấu hình đường dẫn thư viện chia sẻ bằng các tệp trong `/etc/`, chẳng hạn như `ld.so.conf` hoặc các tệp trong thư mục `ld.so.conf.d`.
- Cách định cấu hình đường dẫn thư viện chia sẻ bằng biến môi trường `LD_LIBRARY_PATH`.
- Cách tra cứu các phần phụ thuộc thư viện chia sẻ và thực thi được.

Các lệnh đã được dùng trong bài học này:

`ls`
: Liệt kê nội dung thư mục.

`cat`
: Nối các tệp và in trên đầu ra tiêu chuẩn.

`sudo`
: Yêu cầu siêu người dùng thực thi lệnh với quyền của quản trị viên hệ thống.

`ldconfig`
: Định cấu hình liên kết thời gian chạy trình liên kết động.

`echo`
: Hiển thị giá trị của biến môi trường.

`export`
: Xuất giá trị của biến môi trường sang vỏ con.

`unset`
: Xóa biến môi trường.

`ldd`

In các phần phụ thuộc đối tượng chia sẻ của một chương trình.

`readelf`

Hiển thị thông tin về các tệp ELF (ELF là viết tắt của *executable and linkable format* - tức định dạng thực thi và có thể liên kết).

`objdump`

In thông tin từ tệp đối tượng.

Đáp án Bài tập Hướng dẫn

1. Hãy chia các tên thư viện chia sẻ sau thành các phần của chúng:

Tên Tệp đầy đủ	**Tên Thư viện**	**Hậu tố *so***	**Số phiên bản**
`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. Bạn đã phát triển một phần mềm và muốn thêm một thư mục thư viện chia sẻ mới vào hệ thống của mình (`/opt/lib/mylib`). Bạn đã viết đường dẫn tuyệt đối của nó trong tệp có tên `mylib.conf`.
   - Bạn nên đặt tệp này trong thư mục nào?

     `/etc/ld.so.conf.d`

   - Bạn nên chạy lệnh nào để các thay đổi có hiệu lực hoàn toàn?

     `ldconfig`

3. Bạn sẽ sử dụng lệnh nào để liệt kê các thư viện chia sẻ theo yêu cầu của `kill`?

   `ldd /bin/kill`

Đáp án Bài tập Mở rộng

1. `objdump` là một tiện ích dòng lệnh hiển thị thông tin từ các tệp đối tượng. Hãy kiểm tra xem nó đã được cài đặt trong hệ thống hay chưa với `which objdump`. Nếu chưa thì bạn hãy cài đặt nó ngay bây giờ.
   - Hãy sử dụng `objdump` với tùy chọn `-p` (hoặc `--private-headers`) và `grep` để in các phần phụ thuộc của `glibc`:

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

   - Hãy sử dụng `objdump` với tùy chọn `-p` (hoặc `--private-headers`) và `grep` để in soname của `glibc`:

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

   - Hãy sử dụng `objdump` với tùy chọn `-p` (hoặc `--private-headers`) và `grep` để in các phần phụ thuộc của Bash:

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

102.4 Sử dụng Trình quản lý gói Debian

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 102.4

Khối lượng

3

Các lĩnh vực kiến thức chính

- Cài đặt, nâng cấp và gỡ cài đặt các gói nhị phân Debian.
- Tìm gói chứa các tệp hoặc thư viện cụ thể có thể được cài đặt hoặc không.
- Nhận thông tin gói như phiên bản, nội dung, phần phụ thuộc, tính toàn vẹn của gói và trạng thái cài đặt (gói có được cài đặt hay không).
- Kiến thức về apt.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

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

102.4 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	102 Cài đặt Linux và Quản lý Gói
Mục tiêu:	102.4 Sử dụng Trình Quản lý Gói Debian
Bài:	1 trên 1

Giới thiệu

Rất lâu về trước khi Linux vẫn còn sơ khai, cách phổ biến nhất để phân phối phần mềm là sử dụng một tệp nén (thường là tệp lưu trữ `.tar.gz`) cùng với mã nguồn mà người dùng sẽ tự giải nén và biên dịch.

Tuy nhiên, khi số lượng và độ phức tạp của phần mềm tăng lên, nhu cầu về cách phân phối phần mềm được biên dịch sẵn trở nên rất rõ ràng. Nói cho cùng thì không phải ai cũng có đủ tài nguyên về cả thời gian lẫn vật chất để tự biên dịch các dự án lớn như nhân Linux hoặc Máy chủ X.

Ngay sau đó, các nỗ lực trong việc tiêu chuẩn hóa cách phân phối các "gói" phần mềm này đã tăng lên và các trình quản lý gói đầu tiên đã ra đời. Những công cụ này giúp việc cài đặt, định cấu hình hoặc gỡ bỏ phần mềm khỏi hệ thống trở nên dễ dàng hơn rất nhiều.

Một trong số đó là định dạng gói Debian (`.deb`) và công cụ gói của nó (`dpkg`). Ngày nay, chúng được sử dụng rộng rãi không chỉ trên chính Debian mà còn cả trên các sản phẩm phái sinh của Debian như Ubuntu và những sản phẩm phái sinh từ Ubuntu.

Một công cụ quản lý gói khác cũng phổ biến trên các hệ thống dựa trên Debian là *Công cụ Gói Nâng cao* (`apt`). Công cụ này có thể sắp xếp một cách hợp lý các khía cạnh của việc cài đặt, bảo trì,

gỡ bỏ các gói và khiến cho những tác vụ này trở nên dễ dàng hơn.

Trong bài học này, chúng ta sẽ học cách sử dụng cả `dpkg` và `apt` để lấy, cài đặt, bảo trì và gỡ bỏ phần mềm trên hệ thống Linux dựa trên Debian.

Công cụ Gói Debian (dpkg)

Công cụ *Gói Debian* (`dpkg`) là tiện ích thiết yếu để cài đặt, cấu hình, bảo trì và gỡ bỏ các gói phần mềm trên các hệ thống dựa trên Debian. Thao tác cơ bản nhất là cài đặt gói `.deb` và nó có thể được thực hiện với:

```
# dpkg -i PACKAGENAME
```

Trong đó `PACKAGENAME` là tên của tệp `.deb` cần cài đặt.

Việc nâng cấp gói cũng được xử lý theo cùng một cách. Trước khi cài đặt gói, `dpkg` sẽ kiểm tra xem phiên bản trước đó đã tồn tại trong hệ thống hay chưa. Nếu đã có, gói sẽ được nâng cấp lên phiên bản mới. Nếu không, một bản sao mới sẽ được cài đặt.

Xử lý các Phần Phụ thuộc

Thông thường, một gói có thể sẽ phải phụ thuộc vào các gói khác để có thể hoạt động được như mong muốn. Ví dụ: trình chỉnh sửa hình ảnh có thể cần tới thư viện để mở tệp JPEG hoặc một tiện ích có thể sẽ cần tới bộ công cụ tiện ích như Qt hoặc GTK cho giao diện người dùng của nó.

`dpkg` sẽ kiểm tra xem các phần phụ thuộc đó đã được cài đặt trên hệ thống hay chưa và nếu chưa thì việc cài đặt gói sẽ không thực hiện được. Trong trường hợp này, `dpkg` sẽ liệt kê các gói bị thiếu. Tuy nhiên, nó *không thể* tự xử lý các phần phụ thuộc. Việc tìm các gói `.deb` với các phần phụ thuộc tương ứng và cài đặt chúng là tùy thuộc vào người dùng.

Trong ví dụ bên dưới, người dùng đang cố gắng cài đặt gói trình chỉnh sửa video OpenShot nhưng lại bị thiếu mất một số phần phụ thuộc:

```
# 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
```

Như đã trình bày ở trên, OpenShot phụ thuộc vào các gói `fonts-cantarell`, `python3-openshot`, `python3-pyqt5`, `python3-pyqt5.qtsvg`, `python3-pyqt5.qtwebkit` và `python3-zmq`. Tất cả những gói này cần được cài đặt trước rồi mới có thể cài đặt OpenShot.

Gỡ bỏ Gói

Để xóa một gói, hãy truyền tham số `-r` cho `dpkg`, theo sau là tên gói. Ví dụ: lệnh sau sẽ xóa gói `unrar` khỏi hệ thống:

```
# 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) ...
```

Thao tác xóa cũng sẽ khởi động việc kiểm tra phần phụ thuộc và ta sẽ không thể xóa được gói trừ khi mọi gói khác phụ thuộc vào gói đó cũng bị xóa. Nếu vẫn cố làm như vậy, người dùng sẽ nhận được thông báo lỗi như bên dưới:

```
# 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
```

Chúng ta có thể chuyển nhiều tên gói cho `dpkg -r` để có thể xoá tất cả cùng một lúc.

Khi một gói bị xóa, các tệp cấu hình tương ứng sẽ được để lại trên hệ thống. Nếu muốn xóa *tất cả mọi thứ* được liên kết với gói, hãy sử dụng tùy chọn `-P` (purge - thanh lọc) thay vì `-r`.

NOTE

Ta có thể buộc `dpkg` cài đặt hoặc gỡ bỏ một gói ngay cả khi các phần phụ thuộc không được đáp ứng bằng cách thêm tham số `--force` (như trong `dpkg -i --force PACKAGENAME`). Tuy nhiên, làm như vậy rất có thể sẽ khiến gói đã được cài đặt hoặc thậm chí là hệ thống bị hỏng. *Đừng* sử dụng `--force` trừ khi bạn hoàn toàn chắc chắn về những gì mình đang làm.

Lấy Thông tin Gói

Để lấy thông tin về gói `.deb` (chẳng hạn như phiên bản, kiến trúc, trình bảo trì, phần phụ thuộc, v.v.), hãy sử dụng lệnh `dpkg` với tham số `-I`, theo sau là tên tệp của gói cần kiểm tra:

```
# 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.
```

Liệt kê các Gói đã được cài đặt và Nội dung Gói

Để có được danh sách các gói đã được cài đặt trên hệ thống, hãy sử dụng tùy chọn `--get -selections` như trong `dpkg --get-selections`. Ta cũng có thể nhận danh sách của tất cả các tệp được cài đặt bởi một gói cụ thể bằng cách truyền tham số `-L PACKAGENAME` cho `dpkg` như dưới đây:

```
# 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
```

Tìm ra Gói nào đang sở hữu một Tệp cụ thể

Đôi khi người dùng sẽ cần tìm ra gói nào đang sở hữu một tệp cụ thể trong hệ thống của mình. Chúng ta có thể thực hiện điều này bằng cách sử dụng tiện ích `dpkg-query`, theo sau là tham số `-S` và đường dẫn đến tệp được đề cập:

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

```
unrar: /usr/bin/unrar-nonfree
```

Cấu hình lại các Gói đã được cài đặt

Khi một gói được cài đặt, có một bước cấu hình được gọi là *bước sau cài đặt* mà trong đó, một tệp lệnh sẽ chạy để thiết lập mọi thứ cần thiết để phần mềm có thể chạy (như quyền, vị trí của tệp cấu hình, v.v). Điều này cũng có thể đặt ra một số câu hỏi cho người dùng về việc thiết lập các tùy chọn để chạy phần mềm.

Đôi khi, do tệp cấu hình bị hỏng hoặc không đúng định dạng, người dùng có thể sẽ muốn khôi phục cài đặt của gói về trạng thái "ban đầu". Hoặc có thể họ sẽ muốn thay đổi câu trả lời mà mình đã đưa ra về các câu hỏi cấu hình ban đầu. Để thực hiện việc này, hãy chạy tiện ích `dpkg-reconfigure`, theo sau là tên gói.

Chương trình này sẽ sao lưu các tệp cấu hình cũ, giải nén tệp cấu hình mới vào đúng thư mục và chạy tệp lệnh *sau cài đặt* do gói cung cấp (như thể gói đang được cài đặt lần đầu tiên). Hãy thử cấu hình lại gói `tzdata` bằng ví dụ sau:

```
# dpkg-reconfigure tzdata
```

Công cụ Gói Nâng cao (apt)

Công cụ Gói Nâng cao (APT) là một hệ thống quản lý gói bao gồm một bộ công cụ giúp đơn giản hóa rất nhiều việc cài đặt, nâng cấp, gỡ bỏ và quản lý gói. APT cung cấp các tính năng như khả năng tìm kiếm nâng cao và xử lý tự động các phần phụ thuộc.

APT không phải là một biện pháp "thay thế" cho `dpkg`. Ta có thể coi nó như một "giao diện người dùng" có khả năng hợp lý hóa các hoạt động và lấp đầy các khoảng trống trong chức năng của `dpkg` (chẳng hạn như giải pháp xử lý các phần phụ thuộc).

APT sẽ hoạt động phối hợp với các kho phần mềm chứa các gói có sẵn để cài đặt. Các kho lưu trữ như vậy có thể là một máy chủ cục bộ hoặc máy chủ từ xa, hoặc (ít phổ biến hơn) thậm chí là đĩa CD-ROM.

Các bản phân phối Linux (chẳng hạn như Debian và Ubuntu) duy trì các kho lưu trữ của riêng chúng và các kho lưu trữ khác có thể được các nhà phát triển hoặc nhóm người dùng duy trì để cung cấp phần mềm không có sẵn từ các kho phân phối chính.

Có rất nhiều tiện ích tương tác với APT. Những tiện ích chính là:

`apt-get`
: được sử dụng để tải xuống, cài đặt, nâng cấp hoặc xóa các gói khỏi hệ thống.

`apt-cache`
: được sử dụng để thực hiện các hoạt động như tìm kiếm trong chỉ mục gói.

`apt-file`
: được sử dụng để tìm kiếm các tệp bên trong các gói.

Ngoài ra còn có một tiện ích "thân thiện hơn" có tên đơn giản là `apt`. Nó kết hợp các tùy chọn được sử dụng nhiều nhất là `apt-get` và `apt-cache` trong cùng một tiện ích. Nhiều lệnh dành cho `apt` sẽ giống với các lệnh dành cho `apt-get`; vì vậy, trong nhiều trường hợp, chúng có thể được hoán đổi cho nhau. Tuy nhiên, vì `apt` không được cài đặt trên tất cả các hệ thống, người dùng được khuyến khích tìm hiểu cách sử dụng `apt-get` và `apt-cache`.

NOTE `apt` và `apt-get` có thể yêu cầu kết nối mạng vì các gói và chỉ mục gói có thể sẽ cần được tải xuống từ một máy chủ từ xa.

Cập nhật Chỉ mục Gói

Trước khi cài đặt hoặc nâng cấp phần mềm bằng APT, ta nên cập nhật chỉ mục gói trước để truy xuất thông tin về các gói mới và gói cập nhật. Điều này được thực hiện bằng lệnh `apt-get`, theo sau là tham số `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 Thay vì `apt-get update`, ta cũng có thể sử dụng `apt update`.

Cài đặt và Gỡ bỏ các Gói

Với chỉ mục gói được cập nhật, bây giờ ta đã có thể cài đặt gói. Điều này được thực hiện với `apt-get install`, theo sau là tên của gói cần cài đặt:

```
# 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.
```

Tương tự, để xóa gói, hãy sử dụng `apt-get remove`, theo sau là tên gói:

```
# 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]
```

Hãy lưu ý rằng khi cài đặt hoặc gỡ bỏ các gói, APT sẽ thực hiện xử lý tự động các phần phụ thuộc. Điều này có nghĩa là mọi gói bổ sung cần thiết cho gói người dùng đang cài đặt *cũng sẽ được cài đặt*, và các gói phụ thuộc vào gói đang bị xóa *cũng sẽ bị xóa*. APT sẽ luôn hiển thị danh sách các mục sẽ được cài đặt hoặc gỡ bỏ trước khi hỏi xem người dùng có muốn tiếp tục hay không:

```
# 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]
```

Hãy lưu ý rằng khi một gói đã bị xóa, các tệp cấu hình tương ứng sẽ được để lại trên hệ thống. Để xóa gói *và* bất kỳ một tệp cấu hình nào, hãy sử dụng tham số `purge` thay vì `remove` hoặc tham số `remove` với tùy chọn `--purge`:

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

hoặc

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

TIP Bạn cũng có thể sử dụng `apt install` và `apt remove`.

Sửa lỗi Phần Phụ thuộc bị hỏng

Có thể sẽ có "phần phụ thuộc bị hỏng" trên một hệ thống. Điều này có nghĩa là một hoặc nhiều gói đã được cài đặt phụ thuộc vào các gói khác chưa được cài đặt hoặc không còn tồn tại nữa. Điều này có thể xảy ra do lỗi APT hoặc do gói được cài đặt thủ công.

Để giải quyết vấn đề này, hãy sử dụng lệnh `apt-get install -f`. Lệnh này sẽ cố gắng "sửa chữa" các gói bị hỏng bằng cách cài đặt các phần phụ thuộc bị thiếu, đảm bảo rằng tất cả các gói đều được nhất quán trở lại.

TIP Bạn cũng có thể sử dụng `apt install -f`.

Nâng cấp Gói

APT có thể được sử dụng để tự động nâng cấp mọi gói đã được cài đặt lên phiên bản mới nhất có sẵn từ kho lưu trữ. Điều này có thể được thực hiện bằng lệnh `apt-get upgrade`. Trước khi chạy nó, hãy cập nhật chỉ mục gói bằng `apt-get update` trước tiên:

```
# 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]
```

Phần tóm tắt ở cuối đầu ra sẽ cho ta biết có bao nhiêu gói sẽ được nâng cấp, bao nhiêu gói sẽ được cài đặt, gỡ bỏ hoặc giữ lại, tổng kích thước tải xuống và dung lượng đĩa bổ sung sẽ cần để hoàn tất thao tác. Để hoàn thành nâng cấp, ta chỉ cần trả lời `Y` và đợi `apt-get` hoàn thành nhiệm vụ của nó.

Để nâng cấp một gói duy nhất, chỉ cần chạy `apt-get upgrade`, theo sau là tên gói. Như trong `dpkg`, `apt-get` trước tiên sẽ kiểm tra xem phiên bản trước của gói đã được cài đặt chưa. Nếu đã được cài đặt, gói sẽ được nâng cấp lên phiên bản mới nhất hiện có trong kho lưu trữ. Nếu không, một bản sao mới sẽ được cài đặt.

TIP Bạn cũng có thể sử dụng `apt upgrade` và `apt update`.

Bộ nhớ Đệm Cục bộ

Khi cài đặt hoặc cập nhật một gói, tệp `.deb` tương ứng sẽ được tải xuống thư mục bộ nhớ đệm cục bộ trước khi gói được cài đặt. Theo mặc định, thư mục này là `/var/cache/apt/archives`. Các tệp đã tải xuống một phần sẽ được sao chép vào `/var/cache/apt/archives/partial/`.

Khi cài đặt và nâng cấp các gói, thư mục bộ nhớ đệm có thể sẽ khá lớn. Để lấy lại dung lượng, ta có thể làm trống bộ nhớ đệm bằng cách sử dụng lệnh `apt-get clean`. Thao tác này sẽ xóa nội dung của các thư mục `/var/cache/apt/archives` và `/var/cache/apt/archives/partial/`.

TIP Bạn cũng có thể sử dụng `apt clean`.

Tìm kiếm Gói

Tiện ích `apt-cache` có thể được sử dụng để thực hiện các thao tác trên chỉ mục gói (chẳng hạn như tìm kiếm một gói cụ thể hoặc liệt kê ra gói nào đang chứa một tệp cụ thể).

Để tiến hành tìm kiếm, hãy sử dụng `apt-cache search`, theo sau là mẫu tìm kiếm. Đầu ra sẽ là danh sách tất cả các gói có chứa mẫu, có thể là trong tên gói, mô tả hoặc tệp được cung cấp.

```
# 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
```

Trong ví dụ trên, mục nhập `liblzma5 - XZ-format compression library` dường như không khớp với mẫu. Tuy nhiên, nếu chúng ta hiển thị đầy đủ thông tin (bao gồm cả mô tả) cho gói bằng tham số `show`, ta sẽ tìm thấy mẫu ở đó:

```
# 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: 223533a347dc76a8cc9445cfc6146ec3
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.)
```

Chúng ta có thể sử dụng *biểu thức chính quy* với mẫu tìm kiếm cho phép thực hiện các tìm kiếm mang tính phức tạp (và chính xác) cao. Tuy nhiên, việc này nằm ngoài phạm vi của bài học này.

TIP Bạn cũng có thể sử dụng `apt search` thay vì `apt-cache search` và `apt show` thay vì `apt-cache show`.

Danh sách Nguồn

APT sử dụng danh sách các nguồn để biết được nơi nhận các gói. Danh sách này được lưu trữ trong tệp `sources.list` nằm bên trong thư mục `/etc/apt`. Tệp này có thể được chỉnh sửa trực tiếp bằng một trình soạn thảo văn bản như `vi`, `pico`, `nano` hoặc bằng các tiện ích đồ họa như `aptitude` hoặc `synaptic`.

Một dòng điển hình bên trong `sources.list` sẽ giống như sau:

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

Cú pháp sẽ là loại lưu trữ, URL, bản phân phối và một hoặc nhiều thành phần, trong đó:

Archive type
: Một kho lưu trữ có thể chứa các gói có phần mềm sẵn sàng để chạy (gói nhị phân, loại `deb`) hoặc có mã nguồn của phần mềm này (gói nguồn, mã `deb-src`). Ví dụ trên cung cấp các gói nhị phân.

URL
: URL cho kho lưu trữ.

Distribution
: Tên (hoặc tên mã) cho bản phân phối mà các gói được cung cấp. Một kho lưu trữ có thể lưu trữ các gói dành cho nhiều bản phân phối. Trong ví dụ trên, `disco` là tên mã của Ubuntu 19.04 *Disco Dingo*.

Components
: Mỗi thành phần sẽ đại diện cho một tập hợp các gói. Các thành phần này có thể sẽ khác nhau tuỳ vào tưng bản phân phối Linux. Ví dụ: trên Ubuntu và các sản phẩm phái sinh của nó, chúng sẽ là:

`main`
: chứa các gói mã nguồn mở được hỗ trợ chính thức.

`restricted`
: chứa phần mềm nguồn đóng được hỗ trợ chính thức (chẳng hạn như trình điều khiển thiết bị cho thẻ đồ họa).

`universe`
: chứa phần mềm mã nguồn mở do cộng đồng duy trì.

`multiverse`
: chứa phần mềm không được hỗ trợ có bằng sáng chế hoặc là nguồn đóng.

Trên Debian, các thành phần chính là:

`main`
: bao gồm các gói tuân thủ *Nguyên tắc Phần mềm Tự do của Debian* (DFSG) và không dựa vào phần mềm ngoại vi nguyên tắc này để hoạt động. Các gói được bao gồm ở đây được coi là một phần của bản phân phối Debian.

`contrib`
: chứa các gói tuân theo DFSG nhưng phụ thuộc vào các gói khác không có trong `main`.

`non-free`
: chứa các gói không tuân thủ DFSG.

`security`
: chứa các bản cập nhật bảo mật.

`backports`
: chứa các phiên bản mới hơn của các gói nằm trong `main`.

Chu kỳ phát triển của các phiên bản Debian ổn định là khá dài (khoảng hai năm) và điều này đảm bảo rằng người dùng có thể nhận được các gói cập nhật mới nhất mà không phải sửa đổi kho lưu trữ lõi `chính`.

NOTE Bạn có thể tìm hiểu thêm về *Nguyên tắc Phần mềm Tự do của Debian* tại: https://www.debian.org/social_contract#guidelines

Để thêm một kho lưu trữ mới để lấy các gói, ta chỉ cần thêm dòng tương ứng (thường được cung cấp bởi người duy trì kho lưu trữ) vào cuối `sources.list`, lưu tệp và tải lại chỉ mục gói bằng `apt-get update` . Sau đó, các gói trong kho lưu trữ mới sẽ có sẵn để có thể cài đặt bằng `apt-get`

`install`.

Hãy nhớ rằng các dòng bắt đầu bằng ký tự # sẽ được coi là chú thích và bị bỏ qua.

Thư mục `/etc/apt/sources.list.d`

Bên trong thư mục `/etc/apt/sources.list.d`, ta có thể thêm các tệp có kho lưu trữ bổ sung để APT sử dụng mà không cần sửa đổi tệp `/etc/apt/sources.list` chính. Đây là những tệp văn bản đơn giản và có cùng cú pháp được mô tả ở trên cũng như phần mở rộng tệp `.list`.

Ta có thể thấy được nội dung của tệp có tên `/etc/apt/sources.list.d/buster-backports.list` ở dưới đây:

```
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
```

Liệt kê Nội dung Gói và Tìm kiếm Tệp

Ta có thể sử dụng tiện ích có tên `apt-file` để thực hiện nhiều thao tác hơn trong chỉ mục gói (ví dụ như liệt kê nội dung của một gói hoặc tìm gói chứa một tệp cụ thể). Tiện ích này có thể là chưa được cài đặt mặc định trong mọi hệ thống. Trong trường hợp này, thông thường, chúng ta có thể cài đặt nó bằng cách sử dụng `apt-get`:

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

Sau khi cài đặt, người dùng sẽ cần cập nhật bộ nhớ đệm gói được sử dụng cho `apt-file`:

```
# apt-file update
```

Việc này thường chỉ mất vài giây. Sau đó, `apt-file` đã sẵn sàng để sử dụng.

Để liệt kê nội dung của một gói, hãy sử dụng tham số `list`, theo sau là tên gói:

```
# 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 Bạn cũng có thể sử dụng `apt list` thay vì `apt-file list`.

Chúng ta có thể tìm kiếm tất cả các gói cho một tệp bằng tham số `search`, theo sau là tên tệp. Ví dụ: nếu muốn biết gói nào cung cấp tệp có tên `libSDL2.so`, ta có thể sử dụng:

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

Câu trả lời là gói `libsdl2-dev`, cung cấp tệp `/usr/lib/x86_64-linux-gnu/libSDL2.so`.

Sự khác biệt giữa `apt-file search` và `dpkg-query` là `apt-file search` cũng sẽ tìm kiếm cả các gói đã bị gỡ cài đặt, trong khi `dpkg-query` chỉ có thể hiển thị các tệp thuộc gói đã được cài đặt.

Bài tập Hướng dẫn

1. Lệnh để cài đặt gói có tên `package.deb` bằng cách sử dụng `dpkg` là gì?

2. Bằng cách sử dụng `dpkg-query`, hãy tìm gói chứa tệp có tên `7zr.1.gz`.

3. Bạn có thể xóa gói có tên `unzip` khỏi hệ thống bằng cách sử dụng `dpkg -r unzip` nếu gói `file-roller` phụ thuộc vào gói đó không? Nếu không, cách tốt nhất để làm điều này là gì?

4. Bằng cách sử dụng `apt-file`, làm cách nào để có thể tìm ra gói nào chứa tệp `unrar`?

5. Bằng cách sử dụng `apt-cache`, lệnh hiển thị thông tin cho gói `gimp` là gì?

Bài tập Mở rộng

1. Hãy xét một kho lưu trữ chứa các gói nguồn Debian dành cho bản phân phối `xenial` được lưu trữ tại `http://us.archive.ubuntu.com/ubuntu/` với các gói dành cho thành phần `universe`. Dòng tương ứng sẽ được thêm vào `/etc/apt/sources.list` là gì?

2. Trong khi biên dịch chương trình, bạn gặp phải thông báo lỗi cho biết rằng tệp tiêu đề `zzip-io.h` không có trong hệ thống. Làm cách nào để có thể tìm ra gói nào cung cấp tệp đó?

3. Làm cách nào để có thể bỏ qua cảnh báo phụ thuộc và xóa gói bằng cách sử dụng `dpkg` ngay cả khi có các gói phụ thuộc vào gói đó trong hệ thống?

4. Làm cách nào để có thể nhận thêm thông tin về gói có tên `midori` bằng cách sử dụng `apt`?

5. Trước khi cài đặt hoặc cập nhật các gói bằng `apt`, bạn nên sử dụng lệnh nào để đảm bảo rằng chỉ mục gói được cập nhật?

Tóm tắt

Trong bài học này, chúng ta đã học về:

- Cách sử dụng `dpkg` để cài đặt và gỡ bỏ các gói.
- Cách liệt kê các gói đã được cài đặt và nội dung gói.
- Cách cấu hình lại gói đã được cài đặt.
- `apt` là gì và cách cài đặt, nâng cấp và xóa các gói bằng cách sử dụng nó.
- Cách sử dụng `apt-cache` để tìm kiếm các gói.
- Cách tệp `/etc/apt/sources.list` hoạt động.
- Cách sử dụng `apt-file` để hiển thị nội dung của một gói hoặc cách tìm gói nào chứa một tệp cụ thể.

Các lệnh sau đã được thảo luận trong bài học này:

`dpkg -i`
: Cài đặt một gói hoặc danh sách các gói được phân tách bằng dấu cách.

`dpkg -r`
: Xóa gói hoặc danh sách gói được phân tách bằng dấu cách.

`dpkg -I`
: Kiểm tra một gói, cung cấp thông tin chi tiết về phần mềm mà nó cài đặt và mọi phần phụ thuộc cần thiết.

`dpkg --get-selections`
: Liệt kê mọi gói mà `dpkg` đã được cài đặt trên hệ thống.

`dpkg -L`
: In ra danh sách mọi tệp mà một gói cụ thể cài đặt.

`dpkg-query`
: Với tên tệp được chỉ định, lệnh này sẽ in ra gói đã được cài đặt tệp.

`dpkg-reconfigure`
: Lệnh này sẽ chạy lại tệp lệnh *sau cài đặt* gói để quản trị viên có thể thực hiện các điều chỉnh cấu hình cho quá trình cài đặt của gói.

`apt-get update`

Lệnh này sẽ cập nhật chỉ mục gói cục bộ để khớp với những gì có sẵn trong kho lưu trữ được định cấu hình trong thư mục `/etc/apt/`.

`apt-get install`

Lệnh này sẽ tải xuống một gói từ một kho lưu trữ từ xa và cài đặt nó cùng với các phần phụ thuộc của nó. Nó cũng có thể được sử dụng để cài đặt một gói Debian đã được tải xuống.

`apt-get remove`

Lệnh này sẽ gỡ cài đặt các gói đã chỉ định khỏi hệ thống.

`apt-cache show`

Giống như lệnh `dpkg -I`, lệnh này có thể được sử dụng để hiển thị chi tiết về một gói cụ thể.

`apt-cache search`

Lệnh này sẽ tìm kiếm cơ sở dữ liệu được lưu trong bộ nhớ đệm APT cục bộ của bạn để tìm một gói cụ thể.

`apt-file update`

Lệnh này sẽ cập nhật bộ nhớ đệm gói để lệnh `apt-file` có thể truy vấn nội dung của nó.

`apt-file search`

Lệnh này sẽ tìm kiếm xem một tệp được bao gồm trong gói nào. Một danh sách tất cả các gói chứa mẫu sẽ được trả về.

`apt-file list`

Lệnh này được sử dụng để liệt kê nội dung của một gói giống như lệnh `dpkg -L`.

Đáp án Bài tập Hướng dẫn

1. Lệnh để cài đặt gói có tên `package.deb` bằng cách sử dụng `dpkg` là gì?

   Truyền tham số `-i` cho `dpkg`:

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

2. Bằng cách sử dụng `dpkg-query`, hãy tìm gói chứa tệp có tên `7zr.1.gz`.

   Thêm tham số `-S` vào `dpkg-query`:

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

3. Bạn có thể xóa gói có tên `unzip` khỏi hệ thống bằng cách sử dụng `dpkg -r unzip` nếu gói `file-roller` phụ thuộc vào gói đó không? Nếu không, cách tốt nhất để làm điều này là gì?

   Không. `dpkg` sẽ không xử lý các phần phụ thuộc và sẽ không cho phép bạn xóa gói nếu một gói đã được cài đặt khác phụ thuộc vào gói đó. Trong ví dụ này, trước tiên, bạn có thể xóa `file-roller` (giả sử không có gì phụ thuộc vào nó) và sau đó là `unzip`, hoặc là xóa cả hai cùng lúc với:

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

4. Làm cách nào để có thể tìm ra gói nào chứa tệp `/usr/bin/unrar` bằng cách sử dụng tiện ích `apt-file`?

   Sử dụng tham số `search`, theo sau là đường dẫn (hoặc tên tệp):

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

5. Bằng cách sử dụng `apt-cache`, lệnh hiển thị thông tin cho gói `gimp` là gì?

   Sử dụng tham số `show`, theo sau là tên gói:

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

Đáp án Bài tập Mở rộng

1. Hãy xét một kho lưu trữ chứa các gói nguồn Debian dành cho bản phân phối `xenial` được lưu trữ tại `http://us.archive.ubuntu.com/ubuntu/` với các gói dành cho thành phần `universe`. Dòng tương ứng sẽ được thêm vào `/etc/apt/sources.list` là gì?

   Các gói nguồn thuộc loại `deb-src`; vì vậy, dòng tương ứng phải là:

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

   Dòng này cũng có thể được thêm vào bên trong tệp `.list` trong `/etc/apt/sources.list.d/`. Tên sẽ tùy thuộc vào bạn nhưng nó phải có tính mô tả, chẳng hạn như `xenial_sources.list`.

2. Trong khi biên dịch chương trình, bạn gặp phải thông báo lỗi cho biết rằng tệp tiêu đề `zzip-io.h` không có trong hệ thống. Làm cách nào để có thể tìm ra gói nào cung cấp tệp đó?

   Sử dụng `apt-file search` để tìm gói nào chứa tệp không có trong hệ thống:

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

3. Làm cách nào để có thể bỏ qua cảnh báo phụ thuộc và xóa gói bằng cách sử dụng `dpkg` ngay cả khi có các gói phụ thuộc vào gói đó trong hệ thống?

   Tham số `--force` có thể được sử dụng, nhưng bạn *không bao giờ* nên thực hiện điều này trừ khi bạn biết chính xác mình đang làm gì vì nó có rủi ro lớn (hệ thống sẽ ở trạng thái không nhất quán hoặc "bị hỏng").

4. Làm cách nào để có thể nhận thêm thông tin về gói có tên `midori` bằng cách sử dụng `apt`?

   Sử dụng `apt-cache show`, theo sau là tên gói:

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

5. Trước khi cài đặt hoặc cập nhật các gói bằng `apt`, bạn nên sử dụng lệnh nào để đảm bảo rằng chỉ mục gói được cập nhật?

   `apt-get update` nên được sử dụng. Thao tác này sẽ tải xuống các chỉ mục gói mới nhất từ kho lưu trữ được mô tả trong tệp `/etc/apt/sources.list` hoặc trong thư mục `/etc/apt/sources.list.d/`.

102.5 Sử dụng Trình quản lý gói RPM và YUM

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 102.5

Khối lượng

3

Các lĩnh vực kiến thức chính

- Cài đặt, cài đặt lại, nâng cấp và gỡ bỏ các gói bằng RPM, YUM và Zypper.
- Nhận thông tin về các gói RPM như phiên bản, trạng thái, phần phụ thuộc, tính toàn vẹn và chữ ký.
- Xác định các tệp mà một gói cung cấp cũng như tìm xem một tệp cụ thể đến từ gói nào.
- Kiến thức về dnf.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

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

102.5 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	102 Cài đặt Linux và Quản lý Gói
Mục tiêu:	102.5 Sử dụng Trình Quản lý gói RPM và YUM
Bài:	1 trên 1

Giới thiệu

Rất lâu về trước khi Linux vẫn còn sơ khai, cách phổ biến nhất để phân phối phần mềm là sử dụng một tệp nén (thường là tệp lưu trữ `.tar.gz`) với mã nguồn mà ta sẽ tự giải nén và biên dịch.

Tuy nhiên, khi số lượng và độ phức tạp của phần mềm tăng lên, nhu cầu về cách phân phối phần mềm được biên dịch sẵn trở nên rất rõ ràng. Nói cho cùng thì không phải ai cũng có đủ tài nguyên về cả thời gian lẫn vật chất để tự biên dịch các dự án lớn như nhân Linux hoặc Máy chủ X.

Ngay sau đó, các nỗ lực trong việc tiêu chuẩn hóa cách phân phối các "gói" phần mềm này đã tăng lên và các trình quản lý gói đầu tiên đã ra đời. Những công cụ này giúp việc cài đặt, định cấu hình hoặc gỡ bỏ phần mềm khỏi hệ thống trở nên dễ dàng hơn rất nhiều.

Một trong số đó là *Trình Quản lý Gói RPM* và công cụ tương ứng của nó (`rpm`) được phát triển bởi Red Hat. Ngày nay, chúng được sử dụng rộng rãi không chỉ trên chính Red Hat Enterprise Linux (RHEL) mà còn cả trên các thế hệ sau của nó như Fedora, CentOS, Oracle Linux và các bản phân phối khác (như openSUSE) và thậm chí cả các hệ điều hành khác (như AIX của IBM).

Các công cụ quản lý gói khác phổ biến trên các bản phân phối tương thích với Red Hat là `yum` (YellowDog Updater Modified), `dnf` (Dandified YUM) và `zypper`. Những công cụ này có thể sắp xếp

một cách hợp lý nhiều khía cạnh của việc cài đặt, bảo trì và gỡ bỏ các gói, khiến cho việc quản lý gói trở nên dễ dàng hơn.

Trong bài học này, chúng ta sẽ học cách sử dụng `rpm`, `yum`, `dnf` và `zypper` để lấy, cài đặt, quản lý và gỡ bỏ phần mềm trên hệ thống Linux.

NOTE Mặc dù sử dụng cùng một định dạng gói nhưng giữa các bản phân phối vẫn có những điểm khác biệt cục bộ. Vì vậy, một gói được tạo riêng cho openSUSE có thể sẽ không hoạt động được trên hệ thống RHEL và ngược lại. Khi tìm kiếm các gói, hãy luôn kiểm tra tính tương thích và cố gắng tìm một gói phù hợp nhất với bản phân phối của bạn nếu có thể.

Trình Quản lý Gói RPM (rpm)

Trình Quản lý Gói RPM (`rpm`) là công cụ thiết yếu để quản lý các gói phần mềm trên các hệ thống dựa trên (hoặc được dẫn xuất từ) Red Hat.

Cài đặt, nâng cấp và gỡ bỏ các Gói

Hoạt động cơ bản nhất là cài đặt một gói. Việc này có thể được thực hiện với:

```
# rpm -i PACKAGENAME
```

Trong đó `PACKAGENAME` là tên của gói `.rpm` cần cài đặt.

Nếu đã có phiên bản trước của gói trên hệ thống, ta có thể nâng cấp lên phiên bản mới hơn bằng tham số `-U`:

```
# rpm -U PACKAGENAME
```

Nếu không có phiên bản nào được cài đặt trước đó thì một bản sao mới sẽ được cài đặt. Để tránh điều này và *chỉ* nâng cấp gói *đã được cài đặt*, hãy sử dụng tùy chọn `-F`.

Trong cả hai thao tác, ta có thể thêm tham số `-v` để nhận đầu ra chi tiết (thông tin khác được hiển thị trong quá trình cài đặt) và `-h` để in các dấu thăng (#) dưới dạng hỗ trợ trực quan nhằm theo dõi tiến trình cài đặt. Nhiều tham số có thể được kết hợp thành một; vì vậy, `rpm -i -v -h` cũng giống như `rpm -ivh`.

Để xóa một gói đã được cài đặt, hãy truyền tham số `-e` (như trong “erase”) cho `rpm`, theo sau là tên của gói cần xóa:

```
# rpm -e wget
```

Nếu gói đã được cài đặt phụ thuộc vào gói bị xóa, người dùng sẽ nhận được thông báo lỗi:

```
# rpm -e unzip
error: Failed dependencies:
    /usr/bin/unzip is needed by (installed) file-roller-3.28.1-2.el7.x86_64
```

Để hoàn tất thao tác, trước tiên ta cần xóa các gói phụ thuộc vào gói mình muốn xóa (trong ví dụ trên là `file-roller`). Ta có thể chuyển nhiều tên gói cho `rpm -e` để xóa nhiều gói cùng một lúc.

Xử lý các Phần Phụ thuộc

Thông thường, một gói có thể sẽ phải phụ thuộc vào các gói khác để có thể hoạt động được như mong muốn. Ví dụ: trình chỉnh sửa hình ảnh có thể cần tới thư viện để mở tệp JPEG, hoặc một tiện ích có thể sẽ cần tới bộ công cụ tiện ích như Qt hoặc GTK cho giao diện người dùng của nó.

`rpm` sẽ kiểm tra xem các phần phụ thuộc đó đã được cài đặt trên hệ thống hay chưa và nếu không thì nó sẽ không cài đặt gói. Trong trường hợp này, `rpm` sẽ liệt kê những gì còn thiếu. Tuy nhiên, nó *không thể* tự xử lý các phần phụ thuộc.

Trong ví dụ bên dưới, người dùng đã cố gắng cài đặt gói cho trình chỉnh sửa ảnh GIMP nhưng lại bị thiếu một số phần phụ thuộc:

```
# 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
    libgimpthumb-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
```

Người dùng có thể tìm các gói `.rpm` với các phần phụ thuộc tương ứng và cài đặt chúng. Các trình quản lý gói như `yum`, `zypper` và `dnf` có các công cụ có thể cho biết gói nào cung cấp một tệp cụ thể. Những điều đó sẽ được thảo luận trong phần sau của bài học này.

Liệt kê các Gói đã được cài đặt

Để có danh sách tất cả các gói đã được cài đặt trên hệ thống, hãy sử dụng `rpm -qa` ("qa" có thể được hiểu là query all - tức “truy vấn tất cả”).

```
# 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
[...]
```

Lấy Thông tin Gói

Để nhận thông tin về một gói *đã được cài đặt* (chẳng hạn như số phiên bản, kiến trúc, ngày cài đặt, trình đóng gói, tóm tắt, v.v.), hãy sử dụng `rpm` với các tham số `-qi` ("qi" có thể được hiểu là query info - tức "`truy vấn thông tin `"), theo sau là tên gói. Ví dụ:

```
# 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.

Install the unzip package if you need to list, test or extract files from
a zip archive.
```

Để có danh sách các tệp bên trong một gói *đã được cài đặt*, hãy sử dụng tham số `-ql` ("ql" có thể được hiểu là query list - tức “truy vấn danh sách”), theo sau là tên gói:

```
# 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
```

Nếu muốn lấy thông tin hoặc danh sách tệp từ gói *chưa* được cài đặt, ta chỉ cần thêm tham số `-p` vào các lệnh ở trên, theo sau là tên của tệp RPM (`FILENAME`). Vì vậy, `rpm -qi PACKAGENAME` trở thành `rpm -qip FILENAME` và `rpm -ql PACKAGENAME` sẽ trở thành `rpm -qlp FILENAME` như được minh họa bên dưới.

```
# 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)
```

Tìm ra Gói nào sở hữu một Tệp cụ thể

Để biết gói đã được cài đặt nào sở hữu một tệp cụ thể, hãy sử dụng `-qf` (qf" có thể được hiểu là query file - tức “truy vấn tệp”), theo sau là đường dẫn đầy đủ tới tệp:

```
# rpm -qf /usr/bin/unzip
unzip-6.0-19.el7.x86_64
```

Trong ví dụ trên, tệp `/usr/bin/unzip` thuộc về gói `unzip-6.0-19.el7.x86_64`.

Trình Cập nhật YellowDog đã sửa đổi (YUM)

`yum` ban đầu được phát triển dưới cái tên *Trình Cập nhật Yellow Dog* (YUP) - một công cụ để quản lý gói trên bản phân phối Yellow Dog Linux. Theo thời gian, nó đã được phát triển để quản lý các gói trên các hệ thống dựa trên RPM khác như Fedora, CentOS, Red Hat Enterprise Linux và Oracle Linux.

Về mặt chức năng, nó cũng tương tự như tiện ích `apt` trong các hệ thống dựa trên Debian trên phương diện có thể tìm kiếm, cài đặt, cập nhật, xóa các gói và tự động xử lý các phần phụ thuộc. `yum` có thể được sử dụng để cài đặt một gói duy nhất hoặc để nâng cấp toàn bộ hệ thống cùng một lúc.

Tìm kiếm Gói

Để cài đặt một gói, ta cần biết tên của nó. Đối với điều này, ta có thể thực hiện tìm kiếm với `yum search PATTERN`, trong đó, `PATTERN` là tên của gói cần tìm kiếm. Kết quả sẽ là một danh sách các gói có tên hoặc phần tóm tắt có chứa mẫu tìm kiếm được chỉ định. Ví dụ: nếu cần một tiện ích để xử lý các tệp nén 7Zip (có đuôi `.7z`), chúng ta có thể sử dụng:

```
# yum search 7zip
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
============================ 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.
```

Cài đặt, nâng cấp và gỡ bỏ các Gói

Để cài đặt gói bằng `yum`, hãy sử dụng lệnh `yum install PACKAGENAME`, trong đó, `PACKAGENAME` là tên của gói. `yum` sẽ tìm gói và các phần phụ thuộc tương ứng từ một kho lưu trữ trực tuyến và cài đặt mọi thứ vào hệ thống.

```
# yum install p7zip
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
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]:
```

Để nâng cấp gói đã được cài đặt, hãy sử dụng `yum update PACKAGENAME`, trong đó, `PACKAGENAME` là tên của gói cần nâng cấp. Ví dụ:

```
# yum update wget
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
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]:
```

Nếu bỏ qua tên của gói, ta sẽ thể cập nhật tất cả các gói đang có sẵn các bản cập nhật trên hệ thống.

Để kiểm tra xem có bản cập nhật nào cho một gói cụ thể hay không, hãy sử dụng `yum check-update PACKAGENAME`. Tương tự, nếu bỏ qua tên gói, `yum` sẽ kiểm tra các bản cập nhật cho tất cả các gói đã được cài đặt trên hệ thống.

Để xóa gói đã được cài đặt, hãy sử dụng `yum remove PACKAGENAME`, trong đó, `PACKAGENAME` là tên của gói cần xóa.

Tìm ra Gói nào cung cấp một Tệp cụ thể

Trong ví dụ trước, chúng ta đã thảo luận về việc cài đặt trình chỉnh sửa hình ảnh `gimp` không thành công do các phần phụ thuộc chưa được đáp ứng. Tuy nhiên, `rpm` có thể hiển thị tệp nào bị thiếu nhưng sẽ không liệt kê tên gói cung cấp chúng.

Ví dụ: một trong những phần phụ thuộc bị thiếu là `libgimpui-2.0.so.0`. Để xem gói nào cung cấp nó, ta có thể sử dụng `yum whatprovides`, theo sau là tên của tệp cần tìm kiếm:

```
# 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
```

Câu trả lời là `gimp-libs-2.8.22-1.el7.i686`. Sau đó, ta có thể cài đặt gói bằng lệnh `yum install gimp-libs`.

Điều này cũng khả thi đối với các tệp đã có trong hệ thống. Ví dụ: nếu muốn biết tệp `/etc/hosts` đến từ đâu, ta có thể sử dụng:

```
# 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
```

Câu trả lời là `setup-2.8.71-10.el7.noarch`.

Nhận Thông tin về Gói

Để nhận thông tin về gói, chẳng hạn như phiên bản, kiến trúc, mô tả, kích thước, v.v., hãy sử dụng `yum info PACKAGENAME`, trong đó, `PACKAGENAME` là tên của gói cần lấy thông tin:

```
# 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.
```

Quản lý Kho Phần mềm

Đối với `yum`, "repos" được liệt kê trong thư mục `/etc/yum.repos.d/`. Mỗi kho lưu trữ sẽ được đại diện bởi một tệp `.repo` như `CentOS-Base.repo`.

Người dùng có thể thêm các kho bổ sung bằng cách thêm tệp `.repo` vào thư mục được đề cập ở trên hoặc vào cuối `/etc/yum.conf`. Tuy nhiên, cách được khuyến nghị để thêm hoặc quản lý các kho lưu trữ là sử dụng công cụ `yum-config-manager`.

Để thêm kho lưu trữ, hãy sử dụng tham số `--add-repo`, theo sau là URL tới tệp `.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
```

Để có danh sách tất cả các kho lưu trữ có sẵn, hãy sử dụng `yum repolist all`. Chúng ta sẽ nhận được một đầu ra tương tự như thế này:

```
# yum repolist all
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
repo id                      repo name                    status
updates/7/x86_64             CentOS-7 - Updates           enabled:  2,500
updates-source/7             CentOS-7 - Updates Sources   disabled
```

Các kho lưu trữ "bị vô hiệu hóa" (`disabled`) sẽ bị bỏ qua khi cài đặt hoặc nâng cấp phần mềm. Để kích hoạt hoặc vô hiệu hoá kho lưu trữ, hãy sử dụng tiện ích `yum-config-manager`, theo sau là id của kho lưu trữ.

Trong kết quả ở trên, id của kho lưu trữ đã được hiển thị trên cột đầu tiên (`repo id`) của mỗi

dòng. Ta chỉ sử dụng phần đứng trước dấu `/` đầu tiên; vì vậy, id cho repo `CentOS-7 - Updates` là `updates` chứ không phải `updates/7/x86_64`.

```
# yum-config-manager --disable updates
```

Lệnh trên sẽ vô hiệu hóa repo `updates`. Để kích hoạt lại nó, hãy sử dụng:

```
# yum-config-manager --enable updates
```

NOTE

Yum sẽ lưu trữ các gói đã tải xuống và siêu dữ liệu được liên kết trong một thư mục bộ nhớ đệm (thường là `/var/cache/yum`). Khi hệ thống được nâng cấp và các gói mới được cài đặt, kích thước của bộ nhớ đệm này có thể sẽ tăng lên. Để dọn dẹp bộ nhớ đệm và lấy lại dung lượng đĩa, bạn có thể sử dụng lệnh `yum clean`, tiếp theo là nội dung cần dọn dẹp. Các tham số hữu ích nhất là `packages` (`yum clean yum`) để xóa các gói đã tải xuống và `metadata` (`yum clean metadata`) để xóa siêu dữ liệu được liên kết. Hãy xem trang hướng dẫn dành cho `yum` (nhập `man yum`) để biết thêm thông tin.

DNF

`dnf` là công cụ quản lý gói được sử dụng trên Fedora và là một nhánh của `yum`. Vì vậy, nhiều lệnh và tham số của chúng sẽ tương tự nhau. Phần này sẽ cung cấp cho bạn một cái nhìn tổng quan nhanh về `dnf`.

Tìm kiếm các gói

`dnf search PATTERN`, trong đó, `PATTERN` là phần đang cần được tìm kiếm. Ví dụ: `dnf search unzip` sẽ hiển thị tất cả các gói chứa từ `unzip` trong tên hoặc mô tả.

Nhận thông tin về một gói

`dnf info PACKAGENAME`

Cài đặt gói

`dnf install PACKAGENAME`, trong đó, `PACKAGENAME` là tên của gói cần cài đặt. Bạn có thể tìm thấy tên bằng cách thực hiện tìm kiếm.

Gỡ bỏ gói

`dnf remove PACKAGENAME`

Nâng cấp gói

`dnf upgrade PACKAGENAME` được dùng để chỉ cập nhật một gói. Hãy bỏ qua PACKAGENAME để nâng cấp tất cả các gói trong hệ thống.

Tìm ra gói nào cung cấp một tệp cụ thể

`dnf provides FILENAME`

Lấy danh sách tất cả các gói đã được cài đặt trong hệ thống

`dnf list --installed`

Liệt kê nội dung của một gói

`dnf repoquery -l PACKAGENAME`

NOTE `dnf` có một hệ thống trợ giúp tích hợp sẵn để hiển thị thêm thông tin (chẳng hạn như tham số bổ sung) cho mỗi lệnh. Để sử dụng nó, hãy nhập `dnf help`, theo sau là lệnh (ví dụ như `dnf help install`).

Quản lý Kho Phần mềm

Cũng giống như với `yum` và `zypper`, `dnf` được dùng để làm việc với các kho lưu trữ phần mềm (repos). Mỗi bản phân phối sẽ có một danh sách các kho lưu trữ mặc định và quản trị viên có thể thêm hoặc xóa các kho lưu trữ nếu cần.

Để có danh sách tất cả các kho lưu trữ có sẵn, hãy sử dụng `dnf repolist`. Để chỉ liệt kê các kho lưu trữ đã được kích hoạt, hãy thêm tùy chọn `--enabled` và để chỉ liệt kê các kho lưu trữ đã bị vô hiệu hoá, hãy thêm tùy chọn `--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
```

Để thêm một kho lưu trữ, hãy sử dụng `dnf config-manager --add_repo URL`, trong đó, `URL` là URL đầy đủ của kho lưu trữ. Để kích hoạt một kho lưu trữ, hãy sử dụng `dnf config-manager --set-enabled REPO_ID`.

Tương tự như vậy, để vô hiệu hóa kho lưu trữ, hãy sử dụng `dnf config-manager --set-disabled REPO_ID`. Trong cả hai trường hợp, `REPO_ID` là ID duy nhất cho kho lưu trữ mà ta có

thể nhận được bằng cách sử dụng `dnf repolist`. Các kho lưu trữ đã thêm sẽ được kích hoạt theo mặc định.

Các kho lưu trữ được lưu trữ trong các tệp `.repo` trong thư mục `/etc/yum.repos.d/` với cùng một cú pháp được sử dụng cho `yum`.

Zypper

`zypper` là công cụ quản lý gói được sử dụng trên SUSE Linux và OpenSUSE. Về tính năng, nó cũng tương tự như `apt` và `yum` trong việc cho phép cài đặt, cập nhật và xóa các gói khỏi hệ thống với độ phân giải phụ thuộc tự động.

Cập nhật Chỉ mục Gói

Giống như các công cụ quản lý gói khác, `zypper` được dùng để làm việc với các kho lưu trữ chứa các gói và siêu dữ liệu. Siêu dữ liệu này cần được làm mới theo thời gian để tiện ích biết về các gói mới nhất hiện có. Để làm mới, ta chỉ cần gõ:

```
# 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.
```

`zypper` có tính năng tự động làm mới có thể được kích hoạt trên từng cơ sở kho lưu trữ, nghĩa là một số kho lưu trữ có thể được làm mới tự động trước khi cài đặt gói hoặc truy vấn và các kho lưu trữ khác có thể sẽ cần được làm mới theo cách thủ công. Bạn sẽ sớm tìm hiểu về cách kiểm soát tính năng này.

Tìm kiếm Gói

Để tìm kiếm một gói, hãy sử dụng toán tử `search` (hoặc `se`), theo sau là tên gói:

```
# 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
```

Toán tử tìm kiếm cũng có thể được sử dụng để lấy danh sách tất cả các gói đã được cài đặt trong hệ thống. Để làm như vậy, hãy sử dụng tham số `-i` mà không có tên gói (như trong `zypper se -i`).

Để xem một gói cụ thể đã được cài đặt hay chưa, hãy thêm tên gói vào lệnh trên. Ví dụ: lệnh sau sẽ tìm kiếm trong số các gói đã được cài đặt để xác định tất cả các gói có chứa "firefox" trong tên:

```
# 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
```

Để chỉ tìm kiếm trong số các gói *không được cài đặt*, hãy thêm tham số `-u` vào toán tử `se`.

Cài đặt, nâng cấp và gỡ bỏ các Gói

Để cài đặt gói phần mềm, hãy sử dụng toán tử `install` (hoặc `in`), theo sau là tên gói:

```
# 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]
```

`zypper` cũng có thể được sử dụng để cài đặt gói RPM trên đĩa, đồng thời cố gắng đáp ứng các phần phụ thuộc của nó bằng cách sử dụng các gói từ kho lưu trữ. Để làm như vậy, ta chỉ cần cung cấp đường dẫn đầy đủ đến gói thay vì tên gói (chẳng hạn như `zypper trong /home/john/newpackage.rpm`).

Để cập nhật các gói được cài đặt trên hệ thống, hãy sử dụng `zypper update`. Như trong quá trình cài đặt, thao tác này sẽ hiển thị danh sách các gói sẽ được cài đặt/nâng cấp trước khi hỏi người dùng có muốn tiếp tục hay không.

Nếu chỉ muốn liệt kê các bản cập nhật có sẵn mà không cần cài đặt bất kỳ thứ gì, ta có thể sử dụng `zypper list-updates`.

Để xóa gói, hãy sử dụng toán tử `remove` (hoặc `rm`), theo sau là tên gói:

```
# 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]
```

Hãy nhớ rằng việc xóa một gói cũng sẽ xóa mọi gói khác phụ thuộc vào gói đó. Ví dụ:

```
# 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):
```

Tìm ra Gói nào chứa một Tệp cụ thể

Để xem gói nào đang chứa một tệp cụ thể, hãy sử dụng toán tử tìm kiếm, theo sau là tham số `--provides` và tên của tệp (hoặc đường dẫn đầy đủ đến tệp). Ví dụ: nếu muốn biết gói nào chứa tệp `libgimpmodule-2.0.so.0` trong `/usr/lib64/`, ta sẽ sử dụng:

```
# 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
```

Lấy Thông tin Gói

Để xem siêu dữ liệu nào được liên kết với một gói, hãy sử dụng toán tử `info`, theo sau là tên gói. Lệnh này sẽ cung cấp kho lưu trữ gốc, tên gói, phiên bản, kiến trúc, nhà cung cấp, kích thước đã được cài đặt, việc nó được cài đặt hay không, trạng thái (nếu nó được cập nhật), gói nguồn và mô tả.

```
# 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.
```

Quản lý Kho Phần mềm

`zypper` cũng có thể được sử dụng để quản lý kho phần mềm. Để xem danh sách tất cả các kho lưu trữ hiện được đăng ký trong hệ thống, hãy sử dụng `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
```

Ta có thể xem trong cột `Enabled` để biết rằng kho lưu trữ có đã được kích hoạt hay vô hiệu hoá. Ta cũng có thể thay đổi điều này bằng toán tử `modifyrepo`, theo sau là tham số `-e` (kích hoạt) hoặc `-d` (vô hiệu hoá) và bí danh của kho lưu trữ (cột thứ hai trong đầu ra ở trên).

```
# 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.
```

Ở trên chúng ta đã đề cập đến việc `zypper` có khả năng *tự động làm mới* có thể được kích hoạt trên cơ sở của mỗi kho lưu trữ. Khi được kích hoạt, cờ này sẽ khiến `zypper` chạy thao tác làm mới (giống như chạy `zypper refresh`) trước khi làm việc với kho đã chỉ định. Điều này có thể được kiểm soát bằng các tham số `-f` và `-F` của toán tử `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'.
```

Thêm và Xóa Kho lưu trữ

Để thêm kho lưu trữ phần mềm mới cho `zypper`, hãy sử dụng toán tử `addrepo`, theo sau là URL của kho lưu trữ và tên kho lưu trữ như dưới đây:

```
# 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.
```

Trong khi thêm kho lưu trữ, người dùng có thể kích hoạt cập nhật tự động bằng tham số `-f`. Các kho lưu trữ đã thêm sẽ được kích hoạt theo mặc định, nhưng ta có thể thêm và vô hiệu hóa một kho lưu trữ cùng lúc bằng cách sử dụng tham số `-d`.

Để xóa kho lưu trữ, hãy sử dụng toán tử `removerepo`, theo sau là tên kho lưu trữ (Bí danh). Để xóa kho lưu trữ được thêm vào trong ví dụ trên, lệnh sẽ là:

```
# zypper removerepo packman
Removing repository 'packman' .....................................[done]
Repository 'packman' has been removed.
```

Bài tập Hướng dẫn

1. Bằng cách sử dụng `rpm` trên hệ thống Red Hat Enterprise Linux, bạn sẽ cài đặt gói `file-roller-3.28.1-2.el7.x86_64.rpm` có hiển thị thanh tiến trình trong quá trình cài đặt như thế nào?

2. Bằng cách sử dụng `rpm`, hãy tìm xem gói nào chứa tệp `/etc/redhat-release`.

3. Bạn sẽ sử dụng `yum` như thế nào để kiểm tra các bản cập nhật cho tất cả các gói trong hệ thống?

4. Bằng cách sử dụng `zypper`, bạn sẽ vô hiệu hoá kho lưu trữ có tên là repo-extras như thế nào?

5. Nếu có một tệp `.repo` mô tả một kho lưu trữ mới thì tệp này sẽ được đặt ở đâu để DNF có thể nhận ra nó?

Bài tập Mở rộng

1. Bạn sẽ sử dụng `zypper` như thế nào để tìm ra gói sở hữu tệp `/usr/sbin/swapon`?

2. Làm cách nào để có thể lấy danh sách tất cả các gói đã được cài đặt trong hệ thống bằng cách sử dụng `dnf`?

3. Bằng cách sử dụng `dnf`, lệnh để thêm kho lưu trữ tại `https://www.example.url/home:reponame.repo` vào hệ thống là gì?

4. Làm cách nào để có thể sử dụng `zypper` kiểm tra xem gói `unzip` đã được cài đặt hay chưa?

5. Bằng cách sử dụng `yum`, hãy tìm xem gói nào cung cấp tệp `/bin/wget`.

Tóm tắt

Trong bài học này, chúng ta đã học về:

- Cách sử dụng `rpm` để cài đặt, nâng cấp và gỡ bỏ các gói.
- Cách sử dụng `yum`, `zypper` và `dnf`.
- Làm thế nào để có được thông tin về một gói.
- Làm thế nào để có được một danh sách nội dung các gói.
- Làm cách nào để tìm ra tệp đến từ gói nào.
- Cách liệt kê, thêm bớt, kích hoạt và vô hiệu hoá kho phần mềm.

Các lệnh sau đã được thảo luận trong bài học này:

- `rpm`
- `yum`
- `dnf`
- `zypper`

Đáp án Bài tập Hướng dẫn

1. Bằng cách sử dụng `rpm` trên hệ thống Red Hat Enterprise Linux, bạn sẽ cài đặt gói `file-roller-3.28.1-2.el7.x86_64.rpm` có hiển thị thanh tiến trình trong quá trình cài đặt như thế nào?

   Sử dụng tham số `-i` để cài đặt gói và tùy chọn `-h` để bật "ký tự dấu thăng" hiển thị tiến trình cài đặt. Vì vậy, câu trả lời sẽ là `rpm -ih file-roller-3.28.1-2.el7.x86_64.rpm`.

2. Bằng cách sử dụng `rpm`, hãy tìm xem gói nào chứa tệp `/etc/redhat-release`.

   Bạn đang muốn truy vấn thông tin về một tệp. Vì vậy, hãy sử dụng tham số `-qf`: `rpm -qf /etc/redhat-release`.

3. Bạn sẽ sử dụng `yum` như thế nào để kiểm tra các bản cập nhật cho tất cả các gói trong hệ thống?

   Sử dụng thao tác `check-update` *không* đi kèm tên gói: `yum check-update`.

4. Bằng cách sử dụng `zypper`, bạn sẽ vô hiệu hoá kho lưu trữ có tên là repo-extras như thế nào?

   Sử dụng thao tác `modifyrepo` để thay đổi các tham số của kho và tham số `-d` để vô hiệu hoá nó: `zypper modifyrepo -d repo-extras`.

5. Nếu có một tệp `.repo` mô tả một kho lưu trữ mới thì tệp này sẽ được đặt ở đâu để DNF có thể nhận ra nó?

   Các tệp `.repo` cho DNF phải được đặt ở cùng một vị trí được YUM sử dụng, bên trong `/etc/yum.repos.d/`.

Đáp án Bài tập Mở rộng

1. Bạn sẽ sử dụng `zypper` như thế nào để tìm ra gói sở hữu tệp `/usr/sbin/swapon`?

   Sử dụng toán tử `se` (tìm kiếm) và tham số `--provides`: `zypper se --provides /usr/sbin/swapon`.

2. Làm cách nào để có thể lấy danh sách tất cả các gói đã được cài đặt trong hệ thống bằng cách sử dụng `dnf`?

   Sử dụng toán tử `list`, theo sau là tham số `--installed`: `dnf list --installed`.

3. Bằng cách sử dụng `dnf`, lệnh để thêm kho lưu trữ tại `https://www.example.url/home:reponame.repo` vào hệ thống là gì?

   "configuration change" làm việc với các kho lưu trữ; vì vậy, hãy sử dụng `config-manager` và tham số `--add_repo`: `dnf config-manager --add_repo https://www.example.url/home: reponame.repo`.

4. Làm cách nào để có thể sử dụng `zypper` kiểm tra xem gói `unzip` đã được cài đặt hay chưa?

   Bạn cần thực hiện tìm kiếm (`se`) trên các gói (`-i`) đã được cài đặt: `zypper se -i unzip`.

5. Bằng cách sử dụng `yum`, hãy tìm xem gói nào cung cấp tệp `/bin/wget`.

   Để tìm hiểu kho nào cung cấp một tệp, hãy sử dụng `whatprovides` với tên tệp: `yum whatprovides /bin/wget`.

102.6 Linux với tư cách là Máy khách ảo hóa

Tham khảo các mục tiêu LPI

LPIC-1, Exam 101, Objective 102.6

Khối lượng

1

Các lĩnh vực kiến thức chính

- Hiểu khái niệm chung về máy ảo và vùng chứa
- Hiểu về các thành phần phổ biến của máy ảo trong đám mây IaaS (chẳng hạn như phiên bản điện toán, lưu trữ khối và kết nối mạng)
- Hiểu về các thuộc tính riêng của hệ thống Linux và việc chúng thay đổi khi hệ thống được sao chép hoặc sử dụng làm mẫu
- Hiểu về cách sử dụng hình ảnh hệ thống để triển khai máy ảo, phiên bản đám mây và vùng chứa
- Hiểu về các tiện ích mở rộng của Linux được sử dụng để tích hợp Linux với một sản phẩm ảo hóa
- Kiến thức về cloud-init

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

- Máy ảo
- Vùng chứa Linux
- Vùng chứa ứng dụng
- Trình điều khiển máy khách
- Khóa máy chủ SSH

- Id máy D-Bus

102.6 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	102 Cài đặt Linux và Quản lý Gói
Mục tiêu:	102.6 Linux với tư cách là Máy khách Ảo hóa
Bài:	1 trên 1

Giới thiệu

Một trong những điểm mạnh lớn nhất của Linux là tính linh hoạt của nó. Một khía cạnh của tính linh hoạt này là khả năng sử dụng Linux như một phương tiện lưu trữ các hệ điều hành khác hoặc các ứng dụng riêng lẻ trong một môi trường hoàn toàn biệt lập và an toàn. Bài học này sẽ tập trung vào các khái niệm về ảo hóa (virtualization), công nghệ vùng chứa (container) cùng với một số chi tiết kỹ thuật cần được xem xét khi triển khai một máy ảo trên một nền tảng đám mây.

Tổng quan về Ảo hóa

Ảo hóa là một công nghệ cho phép một nền tảng phần mềm được gọi là *trình giám sát máy ảo*. Phần mềm này chạy các tiến trình có chứa một hệ thống máy tính được mô phỏng đầy đủ. Trình giám sát máy ảo chịu trách nhiệm quản lý tài nguyên của phần cứng vật lý có thể được sử dụng bởi các máy ảo riêng lẻ. Các máy ảo này được gọi là *máy khách* của trình giám sát máy ảo. Một máy ảo có nhiều tính chất của một máy tính vật lý được mô phỏng trong phần mềm (chẳng hạn như BIOS của hệ thống và bộ điều khiển ổ đĩa cứng cứng). Một máy ảo thường sẽ sử dụng hình hình ảnh ổ đĩa cứng được lưu trữ dưới dạng các tệp riêng lẻ và sẽ có quyền truy cập vào RAM và CPU của máy chủ thông qua phần mềm giám sát máy ảo. Trình giám sát máy ảo sẽ phân tách quyền truy cập vào tài nguyên phần cứng của hệ thống máy chủ giữa các khách; do đó, nó cho

phép nhiều hệ điều hành chạy trên một hệ thống máy chủ.

Các trình giám sát máy ảo thường được sử dụng cho Linux bao gồm:

Xen

Xen là một trình giám sát máy ảo max nguồn mở Loại 1, tức là nó không dựa vào một hệ điều hành cơ bản để hoạt động. Một trình giám sát máy ảo loại này được gọi là *trình giám sát máy ảo vật lý* vì máy tính có thể khởi động trực tiếp vào trình giám sát máy ảo.

KVM

Kernel Virtual Machine là một mô-đun nhân Linux dành riêng cho ảo hóa. KVM là một trình giám sát máy ảo của dành cho cả Loại 1 và Loại 2 bởi dù nó cần một hệ điều hành Linux chung để hoạt động, nó vẫn có thể hoạt động như một trình giám sát máy ảo hoàn thiện bằng cách tích hợp với một bản cài đặt Linux đang chạy. Các máy ảo được triển khai với KVM sẽ sử dụng trình nền `libvirt` và các tiện ích phần mềm liên quan để tạo và quản lý.

VirtualBox

Một ứng dụng máy tính để bàn phổ biến giúp việc tạo và quản lý các máy ảo trở nên dễ dàng. Oracle VM VirtualBox là một trình đa nền tảng và sẽ hoạt động trên Linux, macOS và Microsoft Windows. Vì VirtualBox yêu cầu một hệ điều hành cơ bản để chạy nên nó là một trình giám sát máy ảo Loại 2.

Một số trình ảo hóa sẽ cho phép di chuyển động máy ảo. Quá trình di chuyển một máy ảo từ bản cài đặt bộ ảo hóa này sang một bản cài đặt bộ ảo hóa khác được gọi là *di cư* và các kỹ thuật liên quan sẽ khác nhau giữa các lần triển khai bộ ảo hóa. Một số phiên di cư chỉ có thể được thực hiện khi hệ thống máy khách đã tắt hoàn toàn và một số khác lại có thể được thực hiện trong khi máy khách đang chạy (được gọi là *di cư trực tiếp*). Những kỹ thuật như vậy có thể hỗ trợ việc bảo trì cho trình ảo hóa hoặc hỗ trợ khả năng phục hồi của hệ thống khi trình ảo hóa không hoạt động và máy khách có thể được chuyển sang một trình ảo hóa đang hoạt động.

Các loại Máy ảo

Có ba loại máy ảo chính là máy khách *ảo hóa hoàn toàn*, máy khách *ảo hóa song song* và máy khách *lai*.

Ảo hoá hoàn toàn

Tất cả các hướng dẫn mà một hệ điều hành khách dự kiến sẽ thực thi phải có khả năng chạy trong quá trình cài đặt một hệ điều hành được ảo hóa hoàn toàn. Lý do cho điều này là không có trình điều khiển phần mềm bổ sung nào được cài đặt trong máy khách để dịch hướng dẫn sang phần cứng mô phỏng hoặc phần cứng thực. Máy khách được ảo hóa hoàn toàn là khi máy khách (hoặc HardwareVM) đó không biết rằng nó chính là một phiên bản máy ảo đang chạy.

Để kiểu ảo hóa này diễn ra được trên các phần cứng dựa trên x86, các phần mở rộng CPU Intel VT-x hoặc AMD-V phải được kích hoạt trên hệ thống đã được cài đặt trình giám sát máy ảo. Điều này có thể được thực hiện từ menu cấu hình phần sụn BIOS hoặc UEFI.

Ảo hoá song song

Máy khách được ảo hóa song song (hoặc PVM) là một máy mà hệ điều hành khách biết rằng đó là một phiên bản máy ảo đang chạy. Những loại máy khách này sẽ sử dụng một hạt nhân đã được sửa đổi và các trình điều khiển đặc biệt (được gọi là *trình điều khiển khách*) để giúp hệ điều hành khách sử dụng tài nguyên phần mềm và phần cứng của trình giám sát máy ảo. Hiệu suất của máy khách được ảo hóa song song thường tốt hơn hiệu suất của máy khách được ảo hóa hoàn toàn do lợi thế mà các trình điều khiển phần mềm này mang lại.

Ảo hoá lai

Ảo hóa song song và ảo hóa hoàn toàn có thể được kết hợp để cho phép các hệ điều hành chưa được sửa đổi nhận được hiệu suất I/O gần như gốc bằng cách sử dụng trình điều khiển ảo hóa song song trên các hệ điều hành ảo hóa hoàn toàn. Trình điều khiển ảo hóa song song sẽ chứa các trình điều khiển thiết bị lưu trữ và mạng với hiệu suất I/O của mạng và đĩa được nâng cao.

Các nền tảng ảo hóa thường cung cấp các trình điều khiển máy khách được đóng gói cho các hệ điều hành ảo hóa. KVM sử dụng trình điều khiển từ dự án *Virtio*, trong khi Oracle VM VirtualBox sử dụng *Tiện ích mở rộng dành cho Máy khách* có sẵn từ tệp ảnh ISO CD-ROM được cho phép tải xuống.

Ví dụ về Máy ảo `libvirt`

Chúng ta sẽ xem xét một ví dụ về máy ảo được quản lý bởi `libvirt` và sử dụng trình giám sát máy ảo KVM. Một máy ảo thường bao gồm một nhóm tệp, chủ yếu là tệp XML dùng để_xác định_ máy ảo (chẳng hạn như cấu hình phần cứng, kết nối mạng, khả năng hiển thị, v.v.) và tệp hình ảnh ổ đĩa cứng được liên kết có chứa phần cài đặt của hệ điều hành và phần mềm của nó.

Trước tiên, chúng ta hãy bắt đầu kiểm tra một ví dụ về tệp cấu hình XML cho một máy ảo và môi trường mạng của nó:

```
$ 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

Phần `qemu` của đường dẫn thư mục nói đến phần mềm cơ bản mà các máy ảo dựa trên KVM phụ thuộc vào. Dự án QEMU có cung cấp phần mềm cho trình giám sát

> máy ảo mô phỏng các thiết bị phần cứng mà máy ảo sẽ sử dụng (chẳng hạn như bộ điều khiển đĩa, quyền truy cập vào CPU chủ, mô phỏng thẻ mạng, v.v.).

Hãy lưu ý rằng có một thư mục có tên `networks`. Thư mục này có chứa các tệp định nghĩa (cũng sử dụng XML) để tạo cấu hình mạng mà các máy ảo có thể sử dụng. Trình giám sát máy ảo này chỉ sử dụng một mạng và do đó, chỉ có một tệp định nghĩa chứa cấu hình cho phân đoạn mạng ảo mà các hệ thống này sẽ sử dụng.

```
$ 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>
```

Định nghĩa này bao gồm một mạng riêng Hạng C và một thiết bị phần cứng giả lập để hoạt động như một bộ định tuyến cho mạng. Ngoài ra còn có một dải địa chỉ IP để trình giám sát máy ảo sử dụng trong việc triển khai máy chủ DHCP có thể được gán cho các máy ảo sử dụng mạng này. Cấu hình mạng này cũng sử dụng *trình dịch địa chỉ mạng* (NAT) để chuyển tiếp các gói đến các mạng khác (chẳng hạn như mạng LAN của trình giám sát máy ảo).

Bây giờ, chúng ta sẽ cùng chuyển sự chú ý sang tệp định nghĩa máy ảo Red Hat Enterprise Linux 8 (các phần ghi chú đặc biệt sẽ được in đậm):

```
$ 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>
```

Tệp này xác định một số cài đặt phần cứng được sử dụng bởi máy khách này chẳng hạn như dung lượng RAM sẽ được gán cho nó, số lượng lõi CPU từ trình giám sát máy ảo mà máy khách sẽ có quyền truy cập, tệp hình ảnh ổ đĩa cứng được liên kết với máy khách này (trong phần `disk`), khả năng hiển thị của nó (thông qua giao thức SPICE), quyền truy cập của máy khách vào thiết bị USB cũng như đầu vào chuột và bàn phím được mô phỏng.

Ví dụ về Lưu trữ Đĩa Máy ảo

Hình ảnh ổ đĩa cứng của máy ảo này nằm ở `/var/lib/libvirt/images/rhel8`. Đây là hình ảnh đĩa trên trình giám sát máy ảo này:

```
$ sudo ls -lh /var/lib/libvirt/images/rhel8
-rw------- 1 root root 5.5G Oct 25 15:57 /var/lib/libvirt/images/rhel8
```

Kích thước hiện tại của hình ảnh đĩa này chỉ chiếm 5,5 GB dung lượng trên trình giám sát máy ảo. Tuy nhiên, hệ điều hành trong máy khách này nhận thấy một đĩa có kích thước 23,3 GB (bằng chứng là đầu ra của lệnh sau từ bên trong máy ảo đang chạy(:

```
$ lsblk
NAME          MAJ:MIN RM  SIZE RO 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]
```

Điều này là do loại cung cấp đĩa được sử dụng cho máy khách này. Có nhiều loại ảnh đĩa mà máy ảo có thể sử dụng, nhưng hai loại chính là:

COW (Copy-on-write)

Sao chép khi ghi (hay còn được gọi là *phân bố mỏng* hoặc *hình ảnh rải rác*) là phương pháp mà trong đó một tệp đĩa sẽ được tạo với giới hạn kích thước được xác định trước. Kích thước ảnh đĩa sẽ chỉ tăng lên khi dữ liệu mới được ghi vào đĩa. Giống như ví dụ trước, hệ điều hành khách sẽ thấy giới hạn đĩa được xác định trước là 23,3 GB nhưng sẽ chỉ ghi 5,5 GB dữ liệu vào tệp đĩa. Định dạng ảnh đĩa được sử dụng cho máy ảo ví dụ là `qcow2` (một định dạng tệp QEMU COW).

RAW

Loại đĩa *thô* hoặc đĩa *toàn phần* là một tệp mà toàn bộ dung lượng của nó đã được phân bổ từ trước. Ví dụ: tệp hình ảnh đĩa thô 10 GB sẽ tiêu thụ 10 GB dung lượng đĩa thực tế trên trình giám sát máy ảo. Kiểu đĩa này có lợi về hiệu suất vì tất cả dung lượng đĩa cần thiết đều đã tồn tại; vì vậy, trình giám sát máy ảo sẽ chỉ có thể ghi dữ liệu vào đĩa mà không cần mất hiệu suất cho việc theo dõi hình ảnh đĩa để đảm bảo rằng nó chưa đạt đến giới hạn và sẽ mở rộng kích thước của tệp khi dữ liệu mới được ghi vào nó.

Có các nền tảng quản lý ảo hóa khác như *Red Hat Enterprise Virtualization* và *oVirt* có thể sử dụng các đĩa vật lý để hoạt động như các vị trí lưu trữ dự phòng cho một hệ điều hành của máy ảo. Các hệ thống này có thể sử dụng mạng vùng lưu trữ (SAN) hoặc thiết bị lưu trữ gắn mạng (NAS) để ghi dữ liệu của chúng và trình giám sát máy ảo sẽ theo dõi vị trí lưu trữ nào thuộc về máy ảo nào. Các hệ thống lưu trữ này có thể sử dụng các công nghệ như Trình quản lý Ổ phân vùng logic (LVM) để tăng hoặc giảm kích thước lưu trữ đĩa của máy ảo khi cần và để hỗ trợ tạo và quản lý hình ảnh tức thời lưu trữ.

Làm việc với các bản mẫu Máy Ảo

Vì các máy ảo thường chỉ là các tệp chạy trên một trình giám sát máy ảo nên rất dễ để tạo *các bản mẫu* có thể tùy chỉnh cho các tình huống triển khai cụ thể. Thông thường, một máy ảo sẽ có một thiết lập hệ điều hành cơ bản và một số cài đặt cấu hình xác thực được định cấu hình sẵn để dễ dàng khởi chạy hệ thống trong tương lai. Điều này có thể cắt giảm đáng kể lượng thời gian xây dựng một hệ thống mới bằng cách giảm khối lượng của các công việc thường xuyên lặp lại (chẳng hạn như cài đặt gói cơ sở và cài đặt ngôn ngữ).

Bản mẫu máy ảo này sau đó có thể được sao chép sang một hệ thống máy khách mới. Trong

trường hợp này, máy khách mới sẽ được đổi tên, một địa chỉ MAC mới sẽ được tạo cho giao diện mạng của nó và các sửa đổi khác có thể được thực hiện tùy thuộc vào mục đích sử dụng.

ID của máy D-Bus

Nhiều bản cài đặt Linux sẽ sử dụng số nhận dạng máy được tạo khi cài đặt. Số nhận dạng này được gọi là *ID máy D-Bus*. Tuy nhiên, nếu một máy ảo được *nhân bản* được dùng làm mẫu cho các cài đặt máy ảo khác, chúng ta sẽ phải tạo một ID máy D-Bus mới để đảm bảo rằng các tài nguyên hệ thống từ trình giám sát máy ảo được chuyển hướng đến hệ thống máy khách một cách thích hợp.

Lệnh sau có thể được sử dụng để xác thực rằng ID máy D-Bus có tồn tại cho hệ thống đang chạy:

```
$ dbus-uuidgen --ensure
```

Nếu không có thông báo lỗi nào được hiển thị thì tức là hệ thống có ID. Để xem ID máy D-Bus hiện tại, hãy chạy mã sau:

```
$ dbus-uuidgen --get
17f2e0698e844e31b12ccd3f9aa4d94a
```

Chuỗi văn bản hiển thị là số ID hiện tại. Các hệ thống Linux chạy trên cùng một trình giám sát máy ảo không được có cùng một ID máy D-Bus.

ID máy D-Bus nằm ở `/var/lib/dbus/machine-id` và được liên kết tượng trưng với `/etc/machine-id`. Việc thay đổi số ID này trên một hệ thống đang chạy không được khuyến khích vì hệ thống sẽ trở nên không ổn định và có thể xảy ra sự cố. Nếu hai máy ảo có cùng một ID máy D-Bus, hãy làm theo tiến trình bên dưới để tạo một máy mới:

```
$ sudo rm -f /etc/machine-id
$ sudo dbus-uuidgen --ensure=/etc/machine-id
```

Trong trường hợp `/var/lib/dbus/machine-id` không phải là một liên kết tượng trưng tương ứng với `/etc/machine-id` thì `/var/lib/dbus/machine-id` sẽ phải bị xóa.

Triển khai Máy ảo trên Đám mây

Có vô số nhà cung cấp IaaS (*Cơ sở Hạ tầng dưới dạng Dịch vụ*) chạy các hệ thống ảo hóa và có thể triển khai các hình ảnh máy khách ảo cho một tổ chức. Trên thực tế, tất cả các nhà cung cấp này

đều có các công cụ cho phép quản trị viên xây dựng, triển khai và định cấu hình các máy ảo tùy chỉnh dựa trên nhiều bản phân phối Linux. Nhiều công ty trong số đó cũng có sẵn các hệ thống cho phép triển khai và di dời các máy ảo được xây dựng từ bên trong tổ chức của khách hàng.

Khi đánh giá việc triển khai một hệ thống Linux trong môi trường IaaS, có một số yếu tố chính mà quản trị viên nên biết:

Máy ảo điện toán

Nhiều nhà cung cấp đám mây sẽ tính phí sử dụng dựa trên "máy ảo điện toán" hoặc lượng thời gian CPU mà cơ sở hạ tầng dựa trên đám mây của người dùng sẽ sử dụng. Việc lập kế hoạch cẩn thận về lượng thời gian xử lý mà các ứng dụng thực sự cần sẽ giúp duy trì chi phí của giải pháp đám mây ở trong tầm kiểm soát.

Phiên bản điện toán cũng thường đề cập đến số lượng máy ảo được cung cấp trong một môi trường đám mây. Một lần nữa, việc có nhiều phiên bản hệ thống đang chạy cùng một lúc cũng sẽ ảnh hưởng đến mức chi phi mà một tổ chức phải trả cho tổng thời gian CPU.

Lưu trữ Khối

Các nhà cung cấp đám mây cũng có sẵn nhiều cấp độ lưu trữ khối khác nhau cho một tổ chức sử dụng. Một số dịch vụ chỉ đơn giản là lưu trữ mạng dựa trên web cho các tệp và các dịch vụ khác liên quan đến bộ nhớ ngoại vi dành cho máy ảo được cung cấp trên đám mây để sử dụng cho việc lưu trữ tệp.

Chi phí cho các dịch vụ như vậy sẽ thay đổi dựa trên dung lượng lưu trữ được sử dụng và tốc độ lưu trữ trong các trung tâm dữ liệu của nhà cung cấp. Truy cập bộ nhớ nhanh hơn thường sẽ tiêu tốn nhiều chi phí hơn và ngược lại, dữ liệu "ở trạng thái nghỉ" (như trong bộ nhớ lưu trữ) thường sẽ có chi phí rất thấp.

Mạng

Một trong những vấn đề chính khi làm việc với nhà cung cấp giải pháp đám mây là cách thức cấu hình mạng ảo. Nhiều nhà cung cấp IaaS sẽ có một số dạng tiện ích dựa trên web có thể được sử dụng để thiết kế và triển khai các tuyến mạng, mạng con và cấu hình tường lửa khác nhau. Một số thậm chí sẽ cung cấp các giải pháp DNS để có thể chỉ định FQDN (*Tên Miền đủ điều kiện*) có thể được truy cập công khai bởi các hệ thống truy cập internet. Thậm chí còn có các giải pháp "hỗn hợp" có thể kết nối cơ sở hạ tầng mạng tại chỗ và hiện có với cơ sở hạ tầng dựa trên đám mây thông qua phương tiện VPN (*mạng riêng ảo*), từ đó liên kết hai cơ sở hạ tầng lại với nhau.

Truy cập an toàn vào Máy khách trên Đám mây

Phương pháp phổ biến nhất được sử dụng để truy cập một máy khách ảo từ xa trên nền tảng đám

mây là thông qua việc sử dụng phần mềm OpenSSH. Một hệ thống Linux nằm trong đám mây sẽ có máy chủ OpenSSH đang chạy, trong khi quản trị viên sẽ sử dụng ứng dụng khách OpenSSH với các khóa được chia sẻ trước để truy cập từ xa.

Một quản trị viên sẽ chạy lệnh sau:

```
$ ssh-keygen
```

Sau đó là làm theo dấu nhắc để tạo cặp khóa SSH công khai và riêng tư. Khóa riêng tư vẫn còn trên hệ thống cục bộ của quản trị viên (được lưu trữ trong `~/.ssh/`) và khóa công khai sẽ được sao chép vào hệ thống đám mây từ xa. Đây cũng chính là phương pháp mà người dùng sẽ sử dụng khi làm việc với các máy được kết nối trong mạng LAN công ty.

Quản trị viên sẽ chạy lệnh sau:

```
$ ssh-copy-id -i <public_key> user@cloud_server
```

Thao tác này sẽ sao chép khóa SSH công khai từ cặp khóa vừa được tạo sang máy chủ đám mây từ xa. Khóa công khai sẽ được ghi vào tệp `~/.ssh/authorized_keys` của máy chủ đám mây và sẽ đặt các quyền thích hợp trên tệp.

NOTE Nếu chỉ có một tệp khóa công khai trong thư mục `~/.ssh/` thì khóa chuyển `-i` có thể được bỏ qua vì lệnh `ssh-copy-id` sẽ là mặc định cho tệp khóa công khai trong thư mục (thường là tệp kết thúc bằng phần mở rộng `.pub`).

Một số nhà cung cấp dịch vụ đám mây sẽ tự động tạo cặp khóa khi hệ thống Linux mới được cung cấp. Sau đó, quản trị viên sẽ phải tải xuống khóa riêng tư cho hệ thống mới từ nhà cung cấp đám mây và lưu trữ khóa đó trên hệ thống cục bộ của họ. Hãy lưu ý rằng quyền đối với khóa SSH phải là `0600` đối với khóa riêng tư và `0644` đối với khóa công khai.

Định cấu hình trước cho Hệ thống Đám mây

Một công cụ hữu ích giúp đơn giản hóa việc triển khai máy ảo dựa trên đám mây là tiện ích `cloud-init`. Cùng với các tệp cấu hình được liên kết và hình ảnh máy ảo được xác định trước, lệnh này là phương pháp trung lập để triển khai máy khách Linux cho rất nhiều các nhà cung cấp IaaS. Bằng cách sử dụng các tệp văn bản thuần túy YAML (*YAML không phải là Ngôn ngữ Đánh dấu*), quản trị viên có thể định cấu hình trước thiết lập mạng, lựa chọn gói phần mềm, cấu hình khóa SSH, tạo tài khoản người dùng, cài đặt ngôn ngữ cùng với vô số các tùy chọn khác để nhanh chóng tạo các hệ thống mới.

Trong quá trình khởi động ban đầu của hệ thống mới, `cloud-init` sẽ đọc cài đặt từ các tệp cấu hình YAML và áp dụng chúng. Quá trình này chỉ cần áp dụng cho thiết lập ban đầu của hệ thống và sẽ giúp việc triển khai một nhóm hệ thống mới trên nền tảng của nhà cung cấp đám mây trở nên dễ dàng hơn.

Cú pháp tệp YAML được sử dụng với `cloud-init` được gọi là *cloud-config*. Sau đây là một tệp `cloud-config` mẫu:

```
#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
```

Hãy lưu ý rằng ở dòng trên cùng không có khoảng cách giữa ký hiệu băm (#) và thuật ngữ `cloud-config`.

NOTE `cloud-init` không chỉ dành cho máy ảo. Bộ công cụ `cloud-init` cũng có thể được sử dụng để định cấu hình trước các vùng chứa (chẳng hạn như vùng chứa LXD Linux) trước khi triển khai.

Vùng Chứa

Công nghệ vùng chứa cũng tương tự như máy ảo ở một số khía cạnh. Đây là nơi người dùng sẽ có một môi trường biệt lập để dễ dàng triển khai một ứng dụng. Đối với một máy ảo, toàn bộ máy tính sẽ được mô phỏng. Trong khi đó, vùng chứa sẽ chỉ sử dụng vừa đủ các phần mềm để chạy một ứng dụng, từ đó có thể giảm thiểu được chi phí.

Các vùng chứa thường linh hoạt hơn so với máy ảo. Một vùng chứa ứng dụng có thể được di dời từ máy chủ này sang máy chủ khác, giống như một máy ảo có thể được di dời từ trình giám sát máy ảo này sang trình giám sát máy ảo khác. Tuy nhiên, đôi khi một máy ảo sẽ cần phải tắt nguồn trước khi có thể di dời; trong khi đó, đối với vùng chứa, ứng dụng sẽ luôn chạy kể cả trong khi nó đang được di dời. Vùng chứa cũng giúp ta dễ dàng triển khai các phiên bản ứng dụng mới song song với phiên bản hiện có. Khi người dùng đóng phiên của họ với các vùng chứa đang chạy, các vùng chứa này có thể được phần mềm điều phối vùng chứa tự động xóa khỏi hệ thống và được

thay thế bằng phiên bản mới, từ đó giúp giảm thiểu thời gian chết.

NOTE Có rất nhiều công nghệ vùng chứa có sẵn cho Linux (chẳng hạn như *Docker*, *Kubernetes*, *LXD/LXC*, *systemd-nspawn*, *OpenShift*, v.v.). Việc triển khai chính xác gói phần mềm vùng chứa nằm ngoài phạm vi của kỳ thi LPIC-1.

Các vùng chứa sử dụng cơ chế *nhóm kiểm soát* (hay còn được gọi là *cgroups*) trong nhân Linux. Cgroups là một cách để phân vùng các tài nguyên hệ thống như bộ nhớ, thời gian của bộ xử lý cũng như băng thông mạng và đĩa cho một ứng dụng riêng lẻ. Quản trị viên có thể trực tiếp sử dụng cgroups để đặt giới hạn tài nguyên hệ thống trên một ứng dụng, hoặc một nhóm ứng dụng có thể tồn tại trong một cgroup duy nhất. Về bản chất, cùng với việc cung cấp các công cụ giúp dễ dàng quản lý và triển khai các cgroups, đây chính là nhiệm vụ của phần mềm vùng chứa.

NOTE Hiện tại, kỳ thi LPIC-1 không cần tới kiến thức về cgroups. Khái niệm cgroups được đề cập tới ở đây để ứng viên có một số kiến thức cơ bản tối thiểu về cách một ứng dụng được phân tách vì mục đích sử dụng tài nguyên hệ thống.

Bài tập Hướng dẫn

1. Phần mở rộng CPU nào là cần thiết trên nền tảng phần cứng dựa trên x86 chạy các máy khách được ảo hóa hoàn toàn?

2. Một phiên cài đặt máy chủ quan trọng yêu cầu hiệu suất nhanh nhất có thể sẽ sử dụng loại ảo hóa nào?

3. Hai máy ảo đã được sao chép từ cùng một bản mẫu và sử dụng D-Bus đang hoạt động không ổn định. Cả hai đều có tên máy chủ và cài đặt cấu hình mạng riêng biệt. Lệnh nào sẽ được sử dụng để xác định xem mỗi máy ảo có ID Máy D-Bus khác nhau hay không?

Bài tập Mở rộng

1. Hãy chạy lệnh sau để xem hệ thống đã kích hoạt phần mở rộng CPU để chạy máy ảo chưa (kết quả tùy thuộc vào CPU của bạn):

   ```
   grep --color -E "vmx|svm" /proc/cpuinfo
   ```

   Tùy thuộc vào đầu ra, bạn có thể đánh dấu `vmx` (đối với CPU hỗ trợ Intel VT-x) hoặc `svm` (đối với CPU hỗ trợ AMD SVM). Nếu bạn không nhận được kết quả, hãy tham khảo hướng dẫn phần sụn BIOS hoặc UEFI của bạn về cách kích hoạt ảo hóa cho bộ xử lý.

2. Nếu bộ xử lý của bạn hỗ trợ ảo hóa, hãy tìm tài liệu về bản phân phối của bạn để chạy trình giám sát máy ảo KVM.
   - Hãy cài đặt các gói cần thiết để chạy trình giám sát ảo hoá KVM.
   - Nếu đang sử dụng môi trường máy tính để bàn đồ họa, bạn cũng nên cài đặt ứng dụng `virt-manager` - tức giao diện người dùng đồ họa có thể được sử dụng trên bản cài đặt KVM. Điều này sẽ hỗ trợ bạn trong việc cài đặt và quản lý máy ảo.
   - Hãy tải xuống hình ảnh ISO của bản phân phối Linux mà bạn chọn và làm theo tài liệu về bản phân phối để tạo một máy ảo mới bằng cách sử dụng ISO này.

Tóm tắt

Trong bài học này, chúng ta đã đề cập đến các khái niệm cơ bản về máy ảo và vùng chứa cũng như cách sử dụng các công nghệ này với Linux.

Chúng ta đã mô tả ngắn gọn các lệnh sau:

`dbus-uuidgen`

Được sử dụng để xác minh và xem ID DBus của hệ thống.

`ssh-keygen`

Được sử dụng để tạo cặp khóa SSH công khai và riêng tư để sử dụng khi truy cập các hệ thống từ xa dựa trên đám mây.

`ssh-copy-id`

Được sử dụng để sao chép khóa SSH công khai của hệ thống sang hệ thống từ xa nhằm hỗ trợ xác thực từ xa.

`cloud-init`

Được sử dụng để hỗ trợ cấu hình và triển khai các máy ảo và vùng chứa vào môi trường đám mây.

Đáp án Bài tập Hướng dẫn

1. Phần mở rộng CPU nào là cần thiết trên nền tảng phần cứng dựa trên x86 chạy các máy khách được ảo hóa hoàn toàn?

   VT-x cho CPU Intel hoặc AMD-V cho CPU AMD.

2. Một phiên cài đặt máy chủ quan trọng yêu cầu hiệu suất nhanh nhất có thể sẽ sử dụng loại ảo hóa nào?

   Một hệ điều hành sử dụng ảo hóa song song (chẳng hạn như Xen) như một hệ điều hành khách có thể tận dụng tốt hơn các tài nguyên phần cứng có sẵn cho nó thông qua việc sử dụng các trình điều khiển phần mềm được thiết kế để hoạt động với trình giám sát máy ảo.

3. Hai máy ảo đã được sao chép từ cùng một bản mẫu và sử dụng D-Bus đang hoạt động không ổn định. Cả hai đều có tên máy chủ và cài đặt cấu hình mạng riêng biệt. Lệnh nào sẽ được sử dụng để xác định xem mỗi máy ảo có ID Máy D-Bus khác nhau hay không?

   `dbus-uuidgen --get`

Đáp án Bài tập Mở rộng

1. Hãy chạy lệnh sau để xem hệ thống đã kích hoạt phần mở rộng CPU để chạy máy ảo chưa (kết quả tùy thuộc vào CPU của bạn):

   Kết quả sẽ khác nhau tùy thuộc vào CPU mà bạn có. Sau đây là một ví dụ đầu ra từ một máy tính có CPU Intel với các tiện ích mở rộng ảo hóa được kích hoạt trong phần sụn 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 aperfmperf 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
   ```

2. Nếu bộ xử lý của bạn hỗ trợ ảo hóa, hãy tìm tài liệu về bản phân phối của bạn để chạy trình giám sát máy ảo KVM.

   - Hãy cài đặt các gói cần thiết để chạy trình giám sát ảo hoá KVM.

     Kết quả sẽ khác nhau tùy thuộc vào bản phân phối của bạn. Sau đây là một số điểm bắt đầu:

     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 +* Nếu đang sử dụng môi trường máy tính để bàn đồ họa, bạn cũng nên cài đặt ứng dụng `virt-manager` - tức giao diện người dùng đồ họa có thể được sử dụng trên bản cài đặt KVM. Điều này sẽ hỗ trợ bạn trong việc cài đặt và quản lý máy ảo.

   Một lần nữa, điều này sẽ thay đổi tuỳ theo bản phân phối. Một ví dụ sử dụng Ubuntu sẽ như sau:

   +

   ```
   $ sudo apt install virt-manager
   ```

+

- Hãy tải xuống hình ảnh ISO của bản phân phối Linux mà bạn chọn và làm theo tài liệu về bản phân phối để tạo một máy ảo mới bằng cách sử dụng ISO này.

  Nhiệm vụ này có thể được xử lý dễ dàng bởi gói `virt-manager`. Tuy nhiên, một máy ảo có thể được tạo từ dòng lệnh bằng cách sử dụng lệnh `virt-install`. Hãy thử cả hai phương pháp để hiểu cách các máy ảo được triển khai.

Linux Professional Institute

Chủ đề 103: Các lệnh GNU và Unix

103.1 Làm việc trên Dòng lệnh

Tham khảo các mục tiêu LPI

LPIC-1 version 5.0, Exam 101, Objective 103.1

Khối lượng

4

Các lĩnh vực kiến thức chính

- Sử dụng các lệnh vỏ đơn và chuỗi lệnh một dòng để thực hiện các tác vụ cơ bản trên dòng lệnh.
- Sử dụng và sửa đổi môi trường vỏ bao gồm xác định, tham chiếu và xuất các biến môi trường.
- Sử dụng và chỉnh sửa lịch sử lệnh.
- Gọi các lệnh bên trong và bên ngoài đường dẫn đã xác định.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

- `bash`
- `echo`
- `env`
- `export`
- `pwd`
- `set`
- `unset`
- `type`
- `which`

- `man`
- `uname`
- `history`
- `.bash_history`
- Quoting

103.1 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.1 Làm việc trên Dòng lệnh
Bài:	1 trên 2

Giới thiệu

Những người mới làm quen với thế giới quản trị của Linux và vỏ Bash thường sẽ cảm thấy hơi lạc lõng vì không có giao diện GUI thân thiện dễ dùng. Họ đã quen với việc chỉ cần nhấp chuột phải là có quyền truy cập vào các gợi ý trực quan và thông tin theo ngữ cảnh mà các tiện ích quản lý tệp đồ họa cung cấp. Vì vậy, chúng ta phải nhanh chóng tìm hiểu và thành thạo việc sử dụng bộ công cụ dòng lệnh tương đối nhỏ gọn mà qua đó, chúng ta có thể ngay lập tức khai thác toàn bộ dữ liệu được cung cấp bởi GUI cũ và hơn thế nữa.

Lấy Thông tin Hệ thống

Khi nhìn vào dấu nhắc dòng lệnh nhấp nháy, câu hỏi đầu tiên của bạn chắc hẳn sẽ là “Tôi đang ở đâu?”, hoặc chính xác hơn là “Hiện nay tôi đang ở đâu trong hệ thống tệp Linux và nếu giả sử tôi đã tạo một tệp mới thì nó sẽ nằm ở đâu?” Những gì chúng ta muốn biết ở đây là *thư mục công việc hiện tại của mình*, và lệnh `pwd` sẽ cho ta biết điều này:

```
$ pwd
/home/frank
```

Giả sử rằng Frank hiện đang đăng nhập vào hệ thống và anh ấy hiện đang ở trong thư mục chính của mình là `/home/frank/`. Nếu Frank tạo một tệp trống bằng cách sử dụng lệnh `touch` mà không chỉ định bất kỳ vị trí nào khác trong hệ thống tệp, tệp sẽ được tạo tại `/home/frank/`. Việc liệt kê nội dung thư mục với lệnh `ls` sẽ cho chúng ta thấy tệp mới đó:

```
$ touch newfile
$ ls
newfile
```

Bên cạnh vị trí trong hệ thống tệp, thường chúng ta sẽ cần có cả thông tin về hệ thống Linux đang chạy. Điều này có thể bao gồm cả số phát hành chính xác của bản phân phối hoặc là phiên bản nhân Linux hiện đang được tải. Công cụ `uname` là những gì chúng ta cần ở đây, đặc biệt là `uname` sử dụng cùng với tùy chọn `-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
```

Tại đây, `uname` cho biết máy của Frank đã được cài đặt nhân Linux phiên bản 4.18.0 và đang chạy Ubuntu 18.04 trên CPU 64-bit (`x86_64`).

Nhận Thông tin về Lệnh

Chúng ta sẽ thường bắt gặp nhiều tài liệu nói về các lệnh Linux mà mình chưa hề quen thuộc. Bản thân dòng lệnh đã cung cấp tất cả các thông tin hữu ích về chức năng của các lệnh và cách sử dụng chúng một cách hiệu quả. Có lẽ những thông tin hữu ích nhất sẽ được tìm thấy trong các tệp của hệ thống hướng dẫn (`man`).

Theo quy định, các nhà phát triển Linux sẽ viết các tệp `man` và phân phối chúng cùng với các tiện ích mà họ tạo ra. Các tệp `man` là các tài liệu có cấu trúc cao với nội dung được phân chia trực quan theo các tiêu đề phần tiêu chuẩn. Khi nhập `man` với tên của lệnh theo sau, ta sẽ nhận được các thông tin bao gồm tên lệnh, tóm tắt cách sử dụng ngắn gọn, mô tả chi tiết hơn và một số thông tin cơ bản về lịch sử và cấp phép quan trọng. Sau đây là một ví dụ:

```
$ 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)
```

`man` sẽ chỉ hoạt động khi ta đặt cho nó một tên lệnh chính xác. Tuy nhiên, nếu không chắc chắn về tên của lệnh, ta có thể sử dụng lệnh `apropos` để tìm kiếm thông qua tên và mô tả của trang `man`. Ví

dụ, giả sử người dùng không thể nhớ rằng chính `uname` sẽ cung cấp thông tin về phiên bản nhân Linux hiện tại, họ có thể chuyển từ `kernel` thành `apropos`. Chúng ta có thể sẽ nhận được đầu ra gồm nhiều dòng, nhưng chúng đều sẽ bao gồm các thông tin như sau:

```
$ 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
```

Nếu không cần tài liệu đầy đủ của lệnh, ta có thể nhanh chóng lấy dữ liệu cơ bản về lệnh bằng cách sử dụng `type`. Ví dụ này sử dụng `type` để truy vấn bốn lệnh riêng biệt cùng một lúc. Kết quả đã cho ta thấy rằng `cp` ("copy") là một chương trình nằm trong `/bin/cp` và `kill` (thay đổi trạng thái của một tiến trình đang chạy) là một vỏ tích hợp sẵn — nghĩa là nó là một phần thực của vỏ 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
```

Hãy lưu ý rằng, ngoài việc là một lệnh nhị phân thông thường như `cp`, `uname` còn là "hàm băm." Đó là bởi vì Frank gần đây đã sử dụng `uname` và đã được thêm vào bảng băm để tăng hiệu quả của hệ thống và khiến nó dễ truy cập hơn vào lần chạy tiếp theo. Nếu Frank chạy `type uname` sau khi khởi động hệ thống, anh ấy sẽ thấy rằng `type` sẽ lại một lần nữa mô tả `uname` là một lệnh nhị phân thông thường.

NOTE Một cách nhanh hơn để dọn dẹp bảng băm là chạy lệnh `hash -d`.

Đôi khi — đặc biệt là khi làm việc với các tệp lệnh tự động — ta sẽ cần một nguồn thông tin đơn giản hơn về một lệnh. Lệnh `which` mà lệnh `type` trước đây đã truy vết sẽ không trả lại gì ngoài vị trí tuyệt đối của lệnh. Ví dụ này sẽ định vị cả hai lệnh `uname` và `which`.

```
$ which uname which
/bin/uname
/usr/bin/which
```

NOTE Nếu muốn hiển thị thông tin về các lệnh `buildin`, ta có thể sử dụng lệnh `help`.

Sử dụng Lịch sử Lệnh

Mỗi người dùng thường sẽ đều phải nghiên cứu cẩn thận cách sử dụng mỗi lệnh một cách thích hợp và cách để chạy nó thành công cùng với một loạt các tùy chọn và đối số phức tạp. Nhưng vài tuần sau đó, điều gì sẽ xảy ra khi chúng ta cần chạy cùng một lệnh với các tùy chọn và đối số giống nhau nhưng không thể nhớ hết các chi tiết? Thay vì bắt đầu tìm hiểu lại từ đầu, ta có thể khôi phục lệnh ban đầu bằng cách sử dụng lệnh `history`.

`history` sẽ trả về các lệnh gần đây nhất mà ta đã thực hiện theo thứ tự các lệnh gần nhất xuất hiện ở dưới cùng. Ta có thể dễ dàng tìm kiếm thông qua các lệnh đó bằng một chuỗi cụ thể với lệnh `grep`. Ví dụ sau sẽ tìm kiếm bất kỳ lệnh nào có bao gồm văn bản `bash_history`:

```
$ history | grep bash_history
1605 sudo find /home -name ".bash_history" | xargs grep sudo
```

Ở đây chỉ có một lệnh duy nhất được trả về cùng với số thứ tự là 1605.

Về `bash_history`, đây là tên của một tệp ẩn thực mà chúng ta có thể tìm thấy trong thư mục chính của người dùng. Vì nó là một tệp ẩn (được chỉ định bằng dấu chấm đứng trước tên tệp), nó sẽ chỉ được hiển thị bằng cách liệt kê nội dung thư mục sử dụng `ls` với đối số `-a`:

```
$ ls /home/frank
newfile
$ ls -a /home/frank
.  ..  .bash_history  .bash_logout  .bashrc  .profile  .ssh  newfile
```

Trong tệp `.bash_history` sẽ có gì? Hãy tự mình xem và bạn sẽ thấy hàng trăm hàng ngàn lệnh gần đây nhất của mình. Tuy nhiên, có thể bạn sẽ ngạc nhiên khi thấy rằng một số lệnh gần đây *nhất* không có ở đây. Đó là vì mặc dù chúng đã được thêm ngay vào cơ sở dữ liệu `history` động, các bổ sung mới nhất cho lịch sử lệnh sẽ không được ghi vào tệp `.bash_history` cho đến khi người dùng thoát khỏi phiên làm việc.

Ta có thể tận dụng nội dung của `history` để giúp trải nghiệm dòng lệnh trở nên nhanh hơn và hiệu quả hơn bằng cách sử dụng các phím mũi tên lên và xuống trên bàn phím. Nhấn phím lên nhiều lần sẽ điền vào dòng lệnh các lệnh gần đây. Khi đã tìm được lệnh mình muốn thực hiện, ta có thể chạy nó bằng cách nhấn Enter. Điều này giúp việc gọi lại và sửa đổi các lệnh nhiều lần trong phiên vỏ (nếu muốn) trở nên dễ dàng hơn.

Bài tập Hướng dẫn

1. Hãy sử dụng hệ thống `man` để xác định cách yêu cầu `apropos` xuất một lệnh ngắn gọn để kết quả đầu ra chỉ trả về một thông báo sử dụng ngắn gọn và rồi thoát.

2. Hãy sử dụng hệ thống `man` để xác định giấy phép bản quyền nào được gán cho lệnh `grep`.

Bài tập Mở rộng

1. Hãy xác định kiến trúc phần cứng và phiên bản nhân Linux đang được sử dụng trên máy tính của bạn ở định dạng đầu ra dễ đọc.

2. Hãy in hai mươi dòng cuối cùng của cơ sở dữ liệu `history` động và tệp `.bash_history` để so sánh chúng.

3. Hãy sử dụng công cụ `apropos` để xác định trang `man` nơi bạn sẽ tìm thấy lệnh để hiển thị kích thước của thiết bị khối vật lý được đính kèm theo đơn vị byte thay vì megabyte hoặc gigabyte.

Tóm tắt

Trong bài học này, chúng ta đã học về:

- Cách lấy thông tin về vị trí hệ thống tệp và ngăn xếp phần mềm hệ điều hành của bạn.
- Cách tìm trợ giúp cho việc sử dụng lệnh.
- Làm thế nào để xác định vị trí hệ thống tệp và loại nhị phân lệnh.
- Cách tìm và sử dụng lại các lệnh đã thực hiện trước đó.

Các lệnh sau đã được thảo luận trong bài học này:

`pwd`
: In đường dẫn đến thư mục làm việc hiện tại.

`uname`
: In kiến trúc phần cứng của hệ thống, phiên bản nhân Linux, bản phân phối và bản phát hành bản phân phối.

`man`
: Truy cập các tệp trợ giúp ghi lại cách sử dụng lệnh.

`type`
: In vị trí hệ thống tệp và nhập một hoặc nhiều lệnh.

`which`
: In vị trí hệ thống tệp cho một lệnh.

`history`
: In hoặc sử dụng lại các lệnh đã thực hiện trước đó.

Đáp án Bài tập Hướng dẫn

1. Hãy sử dụng hệ thống `man` để xác định cách yêu cầu `apropos` xuất một lệnh ngắn gọn để kết quả đầu ra chỉ trả về một thông báo sử dụng ngắn gọn và rồi thoát.

   Chạy `man apropos` và cuộn xuống qua phần “Options” cho đến đoạn `--usage`.

2. Hãy sử dụng hệ thống `man` để xác định giấy phép bản quyền nào được gán cho lệnh `grep`.

   Chạy `man grep` và cuộn xuống phần “Copyright” của tài liệu. Hãy lưu ý rằng chương trình sử dụng bản quyền từ Free Software Foundation.

Đáp án Bài tập Mở rộng

1. Hãy xác định kiến trúc phần cứng và phiên bản nhân Linux đang được sử dụng trên máy tính của bạn ở định dạng đầu ra dễ đọc.

   Chạy `man uname`, đọc qua phần "Description" và xác định các đối số lệnh sẽ chỉ hiển thị kết quả chính xác mà bạn muốn. Hãy lưu ý cách `-v` cung cấp cho bạn phiên bản hạt nhân và `-i` sẽ cung cấp nền tảng phần cứng.

   ```
   $ man uname
   $ uname -v
   $ uname -i
   ```

2. Hãy in hai mươi dòng cuối cùng của cơ sở dữ liệu `history` động và tệp `.bash_history` để so sánh chúng.

   ```
   $ history 20
   $ tail -n 20 .bash_history
   ```

3. Hãy sử dụng công cụ `apropos` để xác định trang `man` nơi bạn sẽ tìm thấy lệnh để hiển thị kích thước của thiết bị khối vật lý được đính kèm theo đơn vị byte thay vì megabyte hoặc gigabyte.

   Một cách là chạy `apropos` với chuỗi `block`, đọc qua kết quả, lưu ý rằng `lsblk` sẽ liệt kê các thiết bị khối (và do đó có thể là công cụ phù hợp nhất cho nhu cầu của chúng ta), chạy `man lsblk `, cuộn qua phần "Description" và lưu ý rằng `-b` sẽ hiển thị kích thước thiết bị tính bằng byte. Cuối cùng, chạy `lsblk -b` để xem kết quả.

   ```
   $ apropos block
   $ man lsblk
   $ lsblk -b
   ```

Linux Professional Institute

103.1 Bài 2

[[.fronttable,cols="h,"]

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.1 Làm việc trên Dòng lệnh
Bài:	2 trên 2

Giới thiệu

Môi trường hệ điều hành bao gồm các công cụ cơ bản — như vỏ dòng lệnh và đôi khi là GUI — mà ta sẽ cần đến để hoàn thành các tác vụ. Tuy nhiên, môi trường cũng sẽ có một danh mục các phím tắt và giá trị được thiết lập sẵn đi kèm với nó. Bài học này là nơi chúng ta sẽ tìm hiểu về cách liệt kê, gọi và quản lý các giá trị đó.

Tìm các Biến Môi trường

Vậy làm cách nào để chúng ta xác định các giá trị hiện tại cho từng biến môi trường? Có một cách là thông qua lệnh `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/game
s:/snap/bin
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
[...]
```

Ta sẽ nhận được rất nhiều kết quả — nhiều hơn cả trong đoạn trích trên. Nhưng hãy lưu ý mục nhập `PATH` đang chứa các thư mục mà vỏ (và các chương trình khác) sẽ tìm kiếm các chương trình khác mà không cần phải chỉ định một đường dẫn chi tiết. Từ đó, ta có thể chạy một chương trình nhị phân (giả sử là ở trong `/usr/local/bin` từ trong thư mục chính và nó sẽ chạy như thể tệp là cục bộ).

Hãy thay đổi chủ đề một chút. Lệnh `echo` sẽ in ra màn hình bất cứ thứ gì chúng ta yêu cầu. Dù có tin hay không, có nhiều lúc việc sử dụng `echo` để lặp lại điều gì đó theo đúng nghĩa đen sẽ rất hữu ích.

```
$ echo "Hi. How are you?"
Hi. How are you?
```

Nhưng ta cũng có thể thực hiện một tác vụ khác với `echo`. Khi ta cung cấp cho nó tên của một biến môi trường — và nói với nó rằng đây là một biến bằng cách thêm `$` vào trước tên biến — và sau đó, thay vì chỉ in tên của biến, vỏ sẽ mở rộng nó để cung cấp giá trị cho chúng ta. Nếu đang tự hỏi liệu thư mục yêu thích hiện có trong đường dẫn hay không, người dùng có thể nhanh chóng kiểm tra bằng cách chạy nó qua `echo`:

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/sn
ap/bin
```

Tạo các Biến Môi trường mới

Chúng ta có thể thêm các biến tùy chỉnh của riêng mình vào môi trường. Cách đơn giản nhất là sử dụng ký tự `=`. Chuỗi bên trái sẽ là tên của biến mới và chuỗi bên phải sẽ là giá trị của nó. Bây giờ, ta có thể cung cấp tên biến cho `echo` để xác nhận rằng nó đã hoạt động:

```
$ myvar=hello
$ echo $myvar
hello
```

NOTE Hãy lưu ý rằng sẽ không có khoảng trống ở hai bên của dấu bằng trong quá trình

> gán biến.

Nhưng nó có thực sự hoạt động hay không? Hãy nhập `bash` vào cửa sổ dòng lệnh để mở vỏ mới. Vỏ mới này trông giống hệt như vỏ ta vừa sử dụng, nhưng thực chất nó chính là *con* của vỏ ban đầu (tức *vỏ mẹ*). Bây giờ, bên trong vỏ con mới này, hãy thử để `echo` thực hiện công việc của nó theo cách nó đã làm trước đây. Nhưng lại không có gì xảy ra. Lý do là gì?

```
$ bash
$ echo $myvar

$
```

Biến được tạo theo cách chúng ta vừa làm sẽ chỉ khả dụng cục bộ — tức là trong phiên vỏ tại thời điểm đó. Nếu khởi động một vỏ mới — hoặc đóng phiên bằng lệnh `exit` — biến sẽ không tiếp tục hoạt động. Việc nhập `exit` ở đây sẽ đưa ta trở lại vỏ gốc ban đầu, tức là chúng ta đang cần phải đến. Ta có thể chạy `echo $myvar` một lần nữa nếu muốn xác nhận rằng biến vẫn hợp lệ. Bây giờ, hãy nhập `export myvar` để chuyển biến cho bất kỳ vỏ con nào mà ta sẽ mở sau đó. Hãy thử nhập `bash` để có vỏ mới và sau đó nhập `echo`:

```
$ exit
$ export myvar
$ bash
$ echo $myvar
hello
```

Tất cả những điều này nghe có vẻ hơi ngớ ngẩn khi chúng ta đang tạo ra các vỏ không có mục đích thực sự. Tuy vậy, việc hiểu cách các biến vỏ được truyền qua hệ thống sẽ trở nên rất quan trọng khi chúng ta bắt đầu viết các tệp lệnh quan trọng.

Xóa các Biến Môi trường

Làm thế nào để dọn sạch tất cả các biến tạm thời mà mình đã tạo? Một cách đơn giản là đóng vỏ gốc — hoặc khởi động lại máy tính. Nhưng cũng có nhiều cách đơn giản hơn, chẳng hạn như `unset`. Việc nhập `unset` (không có `$`) sẽ loại bỏ biến. `echo` sẽ chứng minh điều đó ngay bây giờ.

```
$ unset myvar
$ echo $myvar

$
```

Nếu đã có lệnh `unset` thì đương nhiên sẽ phải có một lệnh `set` đi cùng với nó. Việc chạy riêng lệnh `set` sẽ hiển thị nhiều đầu ra, nhưng nó thực sự không khác mấy so với những gì `env` đã cung cấp. Hãy nhìn vào dòng đầu ra đầu tiên khi lọc `PATH`:

```
$ set | grep PATH
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/game
s:/snap/bin
[...]
```

Sự khác biệt giữa `set` và `env` là gì? Đối với mục đích của chúng ta, việc trọng yếu là `set` sẽ xuất ra tất cả các biến và hàm. Hãy cùng minh họa điều này. Chúng ta sẽ tạo một biến mới gọi là `mynewvar` và sau đó xác nhận rằng nó đang ở đó:

```
$ mynewvar=goodbye
$ echo $mynewvar
goodbye
```

Bây giờ, việc chạy `env` trong khi sử dụng `grep` để lọc chuỗi `mynewvar` sẽ không hiển thị bất kỳ đầu ra nào. Nhưng nếu chạy `set` theo cùng một cách thì ta sẽ thấy được biến cục bộ.

```
$ env | grep mynewvar

$ set | grep mynewvar
mynewvar=goodbye
```

Trích dẫn để thoát các Ký tự Đặc biệt

Bây giờ là thời điểm tốt nhất để giới thiệu tới bạn các vấn đề về ký tự đặc biệt. Các ký tự chữ và số (a-z và 0-9) thường sẽ được Bash đọc theo nghĩa đen. Nếu muốn tạo một tệp mới có tên `myfile`, ta chỉ cần nhập `touch`, theo sau là `myfile` và Bash sẽ biết phải làm gì với nó. Nhưng nếu muốn bao gồm một ký tự đặc biệt trong tên tệp của mình, chúng ta sẽ cần thực hiện thêm một số thao tác.

Để minh họa điều này, chúng ta sẽ nhập `touch` và theo sau là tiêu đề: `my big file`. Vấn đề ở đây là có hai khoảng trắng giữa các từ mà Bash sẽ diễn giải. Mặc dù về mặt kỹ thuật thì một khoảng trắng không được coi là một "ký tự" nhưng nó lại giống như một ký tự bởi Bash sẽ không đọc nó theo nghĩa đen. Nếu liệt kê nội dung của thư mục hiện tại, thay vì một tệp có tên `my big file`, chúng ta sẽ thấy ba tệp có tên tương ứng là `my`, `big` và `file`. Đó là bởi vì Bash nghĩ rằng người dùng muốn tạo nhiều tệp với các từ được liệt kê:

```
$ touch my big file
$ ls
my big file
```

Các khoảng trắng cũng sẽ được diễn giải theo cách này nếu chúng ta muốn xóa (`rm`) cả ba tệp trong một lệnh:

```
$ rm my big file
```

Bây giờ, chúng ta hãy thử thực hiện đúng cách. Hãy nhập `touch` và ba phần tên tệp, nhưng lần này, hãy đặt tên tệp trong dấu trích dẫn kép. Bây giờ thì nó đã hoạt động đúng như mong đợi. Việc liệt kê nội dung thư mục sẽ hiển thị cho ta một tệp có tên chuẩn xác.

```
$ touch "my big file"
$ ls
'my big file'
```

Có nhiều cách khác để có được kết quả tương tự. Ví dụ, dấu trích dẫn đơn cũng hoạt động giống như dấu trích dẫn kép (hãy lưu ý rằng dấu trích dẫn đơn sẽ giữ nguyên giá trị theo nghĩa đen của tất cả các ký tự, trong khi dấu trích dẫn kép sẽ giữ nguyên tất cả các ký tự *ngoại trừ* `$`, `` ` ``, `\` và, trong một số trường hợp nhất định, `!`.)

```
$ rm 'my big file'
```

Nếu ta đặt trước mỗi ký tự đặc biệt một dấu gạch chéo ngược, nó sẽ "thoát" đi tính đặc biệt của ký tự và khiến Bash đọc nó theo đúng nghĩa đen.

```
$ touch my\ big\ file
```

Bài tập Hướng dẫn

1. Hãy sử dụng lệnh `export` để thêm một thư mục mới vào đường dẫn của bạn (thư mục này sẽ không tồn tại khi khởi động lại).

2. Hãy sử dụng lệnh `unset` để xóa biến `PATH`. Hãy thử chạy một lệnh (như `sudo cat /etc/shadow`) bằng cách sử dụng `sudo`. Chuyện gì đã xảy ra? Tại sao? (Thoát khỏi vỏ sẽ đưa bạn trở lại trạng thái ban đầu.)

Bài tập Mở rộng

1. Hãy tìm kiếm trên internet và khám phá danh sách đầy đủ các ký tự đặc biệt.
2. Hãy thử chạy các lệnh bằng cách sử dụng các chuỗi gồm các ký tự đặc biệt và sử dụng các phương pháp khác nhau để thoát chúng. Có sự khác biệt giữa cách thức hoạt động của các phương pháp đó không?

Tóm tắt

Trong bài học này, chúng ta đã học về:

- Cách xác định các biến môi trường của hệ thống.
- Cách tạo các biến môi trường của riêng bạn và xuất chúng sang các vỏ khác.
- Cách loại bỏ các biến môi trường và cách sử dụng cả hai lệnh `env` và `set`.
- Làm cách nào để thoát các ký tự đặc biệt để Bash đọc chúng theo nghĩa đen.

Các lệnh sau đã được thảo luận trong bài học này:

`echo`
: In các chuỗi và biến đầu vào.

`env`
: Hiểu và sửa đổi các biến môi trường.

`export`
: Truyền một biến môi trường cho vỏ con.

`unset`
: Bỏ thiết lập giá trị, thuộc tính của các biến vỏ và hàm.

Đáp án Bài tập Hướng dẫn

1. Hãy sử dụng lệnh `export` để thêm thư mục mới vào đường dẫn của bạn (thư mục này sẽ không tồn tại khi khởi động lại).

   Bạn có thể tạm thời thêm một thư mục mới (có thể là `myfiles` nằm trong thư mục chính) vào đường dẫn bằng cách sử dụng `export PATH="/home/yourname/myfiles:$PATH"`. Hãy tạo một tệp lệnh đơn giản trong thư mục `myfiles/`, làm cho nó có thể thực thi được và thử chạy nó từ một thư mục khác. Các lệnh này sẽ giả định bạn đang ở trong thư mục chính có chứa thư mục tên `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. Hãy sử dụng lệnh `unset` để xóa biến `PATH`. Hãy thử chạy một lệnh (như `sudo cat /etc/shadow`) bằng cách sử dụng `sudo`. Chuyện gì đã xảy ra? Tại sao? (Thoát khỏi vỏ sẽ đưa bạn trở lại trạng thái ban đầu.)

   Nhập `unset PATH` sẽ xóa cài đặt đường dẫn hiện tại. Việc cố gọi một chương trình nhị phân mà không có địa chỉ tuyệt đối sẽ không thành công. Vì lý do đó, việc cố gắng chạy một lệnh bằng sudo (bản thân nó là một chương trình nhị phân nằm trong `/usr/bin/sudo`) sẽ không thành công — trừ khi bạn có chỉ định vị trí tuyệt đối (như trong: `/usr/bin/sudo /bin/cat /etc/shadow`). Bạn có thể đặt lại `PATH` của mình bằng cách sử dụng `export` hoặc đơn giản là thoát khỏi vỏ.

Đáp án Bài tập Mở rộng

1. Hãy tìm kiếm trên internet và khám phá danh sách đầy đủ các ký tự đặc biệt.

   Đây là danh sách các ký tự đặc biệt: `& ; | * ? " ' [ ] ( ) $ < > { } # / \ ! ~`.

2. Hãy thử chạy các lệnh bằng cách sử dụng các chuỗi gồm các ký tự đặc biệt và sử dụng các phương pháp khác nhau để thoát chúng. Có sự khác biệt giữa cách thức hoạt động của các phương pháp đó không?

   Việc thoát bằng cách sử dụng các ký tự `"` sẽ bảo toàn các giá trị đặc biệt của ký hiệu đô la ($), dấu đánh dấu ngược `()` và dấu gạch chéo ngược `(\)`. Mặt khác, việc thoát bằng cách sử dụng ký tự `'` sẽ hiển thị *tất cả* các ký tự dưới dạng chữ.

```
$ echo "$mynewvar"
goodbye
$ echo '$mynewvar'
$mynewvar
```

103.2 Xử lý luồng văn bản bằng Bộ lọc

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 103.2

Khối lượng

2

Các lĩnh vực kiến thức chính

- Gửi tệp văn bản và luồng đầu ra thông qua các bộ lọc tiện ích văn bản để sửa đổi đầu ra bằng cách sử dụng các lệnh UNIX tiêu chuẩn có trong gói textutils GNU.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

- `bzcat`
- `cat`
- `cut`
- `head`
- `less`
- `md5sum`
- `nl`
- `od`
- `paste`
- `sed`
- `sha256sum`
- `sha512sum`
- `sort`

- `split`
- `tail`
- `tr`
- `uniq`
- `wc`
- `xzcat`
- `zcat`

103.2 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.2 Xử lý Luồng Văn bản bằng Bộ Lọc
Bài:	1 trên 1

Giới thiệu

Xử lý văn bản là một phần chính trong công việc của mọi quản trị viên hệ thống. Doug McIlroy, một thành viên của nhóm phát triển Unix đời đầu, đã tóm tắt triết lý của Unix và nói rằng (bên cạnh nhiều điều quan trọng khác): “Viết chương trình là để xử lý các luồng văn bản, bởi vì nó chính là một giao diện phổ quát trên phạm vi toàn cầu.” Linux được lấy cảm hứng từ hệ điều hành Unix và luôn nhất quán áp dụng triết lý của Unix. Vì vậy, quản trị viên phải lường trước được việc có rất nhiều công cụ thao tác văn bản trong một bản phân phối Linux.

Đánh giá nhanh về việc Chuyển hướng và Các Đường ống

Những điều sau cũng là triết lý của Unix:

- Viết chương trình để làm một việc và làm tốt việc đó.
- Viết chương trình để có thể làm việc cùng nhau.

Để các chương trình có thể hoạt động cùng nhau, phương thức phổ biến nhất là thông qua việc *dẫn ống* và *chuyển hướng*. Gần như tất cả các chương trình xử lý văn bản của chúng ta đều sẽ lấy văn bản từ đầu vào tiêu chuẩn (*stdin*), xuất nó thành đầu ra tiêu chuẩn (*stdout*) và gửi lỗi cuối

cùng đến đầu ra lỗi tiêu chuẩn (*stderr*). Trừ khi có chỉ định khác, đầu vào tiêu chuẩn sẽ chính là những gì ta nhập trên bàn phím (chương trình sẽ đọc nó sau khi người dùng nhấn phím Enter). Tương tự, đầu ra tiêu chuẩn và các lỗi sẽ được hiển thị trên màn hình cửa sổ dòng lệnh. Hãy cùng xem tiến trình này thực sự hoạt động như thế nào.

Trong cửa sổ dòng lệnh, hãy nhập `cat` rồi nhấn phím Enter. Sau đó, hãy gõ một phần văn bản ngẫu nhiên.

```
$ 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
```

Để biết thêm thông tin về lệnh `cat` (thuật ngữ này xuất phát từ “concatenate” - nối liền), hãy tham khảo các trang hướng dẫn.

NOTE

Nếu đang làm việc trên một bản cài đặt máy chủ Linux đơn giản, một số lệnh như `info` và `less` có thể sẽ không khả dụng. Trong trường hợp đó, hãy cài đặt các công cụ này theo tiến trình thích hợp trong hệ thống như đã được mô tả trong các bài học tương ứng.

Như đã trình bày ở trên, nếu không có chỉ định nào khác, `cat` sẽ đọc từ đầu vào tiêu chuẩn (bất cứ thứ gì người dùng nhập) và xuất bất cứ thứ gì nó đọc được ra cửa sổ dòng lệnh (đầu ra tiêu chuẩn của nó).

Bây giờ, hãy thử như sau:

```
$ 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
```

`>` (dấu lớn hơn) yêu cầu `cat` hướng đầu ra của nó tới tệp `mytextfile` mà không phải đầu ra tiêu chuẩn. Bây giờ, hãy thử như sau:

```
$ 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
```

Điều này có tác dụng sao chép `mytextfile` sang `mynewtextfile`. Ta có thể xác minh rằng hai tệp này có cùng nội dung bằng lệnh `diff`:

```
$ diff mynewtextfile mytextfile
```

Vì đầu ra không có gì, các tệp đều được coi ngang bằng nhau. Bây giờ, hãy thử dùng toán tử chuyển hướng nối (`>>`):

```
$ echo 'This is my new line' >> mynewtextfile
$ diff mynewtextfile mytextfile
4d3
< This is my new line
```

Cho đến nay, chúng ta đã sử dụng việc chuyển hướng để tạo và thao tác với các tệp. Chúng ta cũng có thể sử dụng các đường ống (được biểu thị bằng ký hiệu `|`) để chuyển hướng đầu ra của chương trình này sang một chương trình khác. Hãy cùng tìm các dòng có từ "this":

```
$ 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
```

Bây giờ, chúng ta đã dẫn đầu ra của `cat` sang một lệnh khác là `grep`. Hãy lưu ý khi bỏ qua chức năng phân biệt chữ hoa chữ thường (sử dụng tùy chọn `-i`), kết quả nhận được sẽ có thêm một

dòng bổ sung.

Xử lý Luồng Văn bản

Đọc Tệp Nén

Chúng ta sẽ tạo một tệp có tên `ftu.txt` và chứa một danh sách các lệnh sau:

```
bzcat
cat
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
xzcat
zcat
```

Bây giờ, chúng ta sẽ sử dụng lệnh `grep` để in tất cả các dòng chứa chuỗi `cat`:

```
$ cat ftu.txt | grep cat
bzcat
cat
xzcat
zcat
```

Một cách khác để lấy thông tin này là sử dụng lệnh `grep` để lọc văn bản trực tiếp mà không cần sử dụng một ứng dụng khác để gửi dòng văn bản tới `stdout`.

```
$ grep cat ftu.txt
```

```
bzcat
cat
xzcat
zcat
```

NOTE Hãy nhớ rằng có rất nhiều cách để thực hiện cùng một tác vụ bằng Linux.

Có các lệnh khác để xử lý tệp nén (`bzcat` cho tệp nén `bzip`, `xzcat` cho tệp nén `xz` và `zcat` cho tệp nén `gzip`) và mỗi lệnh đều được sử dụng để xem nội dung của tệp nén dựa trên thuật toán nén được sử dụng.

Sau khi xác minh tệp mới tạo `ftu.txt` là tệp duy nhất trong thư mục, hãy tạo một bản nén `gzip` của tệp:

```
$ ls ftu*
ftu.txt

$ gzip ftu.txt
$ ls ftu*
ftu.txt.gz
```

Tiếp theo, hãy sử dụng lệnh `zcat` để xem nội dung của tệp nén 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
```

Hãy lưu ý rằng `gzip` sẽ nén `ftu.txt` thành `ftu.txt.gz` và xóa tệp gốc. Theo mặc định, sẽ không có đầu ra nào từ lệnh `gzip` được hiển thị. Tuy nhiên, nếu muốn `gzip` cho biết nó đang làm gì, hãy sử dụng tùy chọn `-v` (verbose) để có được đầu ra "chi tiết".

Xem Tệp trong Pager

Ta đã biết rằng `cat` sẽ nối một tệp với đầu ra tiêu chuẩn (một khi tệp được cung cấp sau lệnh). Tệp `/var/log/syslog` là nơi hệ thống Linux lưu trữ mọi sự kiện quan trọng đang diễn ra trong hệ thống. Hãy sử dụng lệnh `sudo` để nâng đặc quyền và đọc tệp `/var/log/syslog`:

```
$ sudo cat /var/log/syslog
```

...ta sẽ thấy các tin nhắn cuộn rất nhanh trong cửa sổ dòng lệnh. Ta có thể dẫn đầu ra sang chương trình `less` để kết quả được phân trang. Bằng cách sử dụng `less`, ta có thể sử dụng các phím mũi tên để điều hướng thông qua đầu ra và cũng có thể sử dụng các lệnh giống `vi` để điều hướng và tìm kiếm trong toàn bộ văn bản.

Tuy nhiên, thay vì dẫn lệnh `cat` vào một chương trình phân trang, sẽ thực tế hơn nếu ta trực tiếp sử dụng chương trình phân trang:

```
$ sudo less /var/log/syslog
... (output omitted for clarity)
```

Lấy một phần của Tệp Văn bản

Nếu chỉ cần xem lại phần đầu hoặc phần cuối của tệp, ta có thể sử dụng nhiều phương pháp khác. Lệnh `head` được sử dụng để đọc mười dòng đầu tiên của tệp theo mặc định và lệnh `tail` được sử dụng để đọc mười dòng cuối cùng của tệp theo mặc định. Bây giờ, hãy thử:

```
$ 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.
```

Để giúp minh họa số dòng được hiển thị, chúng ta có thể dẫn đầu ra của lệnh `head` thành lệnh `nl`. Lệnh này sẽ hiển thị số dòng văn bản được truyền vào lệnh:

```
$ 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)
```

Và chúng ta có thể làm tương tự bằng cách dẫn đầu ra của lệnh `tail` sang lệnh `wc`. Lệnh này theo

mặc định sẽ đếm số từ trong tài liệu và sử dụng khóa chuyển `-l` để in ra số dòng văn bản mà lệnh đã đọc:

```
$ sudo tail /var/log/syslog | wc -l
10
```

Nếu quản trị viên cần xem lại nhiều hơn (hoặc ít hơn) phần đầu hoặc phần cuối của tệp, tùy chọn `-n` có thể được sử dụng để giới hạn đầu ra của lệnh:

```
$ 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
```

Khái niệm cơ bản về sed - Trình chỉnh sửa Luồng

Chúng ta hãy xem các tệp, thuật ngữ và tiện ích khác không có `cat` trong tên của chúng. Chúng ta có thể làm điều này bằng cách truyền tùy chọn `-v` cho `grep` để lệnh chỉ xuất ra các dòng không

chứa `cat`:

```
$ zcat ftu.txt.gz | grep -v cat
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
```

Hầu hết những gì chúng ta có thể làm với `grep` thì chúng ta cũng có thể làm với `sed` — trình chỉnh sửa luồng — bao gồm việc lọc và chuyển đổi văn bản (như được nêu trong trang hướng dẫn của `sed`). Trước tiên, chúng ta sẽ khôi phục tệp `ftu.txt` của mình bằng cách giải nén kho lưu trữ tệp `gzip`:

```
$ gunzip ftu.txt.gz
$ ls ftu*
ftu.txt
```

Bây giờ, chúng ta có thể sử dụng `sed` để chỉ liệt kê các dòng chứa chuỗi `cat`:

```
$ sed -n /cat/p < ftu.txt
bzcat
cat
xzcat
zcat
```

Chúng ta đã sử dụng dấu nhỏ hơn `<` để hướng nội dung của tệp `ftu.txt into` vào lệnh `sed`. Từ được đặt giữa các dấu gạch chéo (tức `/cat/`) là thuật ngữ mà chúng ta đang tìm kiếm. Tùy chọn `-n` sẽ ra lệnh cho `sed` không tạo đầu ra (trừ khi các kết quả sau này được lệnh `p` hướng dẫn). Hãy thử

chạy cùng lệnh này mà không có tùy chọn `-n` để xem điều gì sẽ xảy ra. Sau đó, hãy thử:

```
$ sed /cat/d < ftu.txt
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
split
tail
tr
uniq
wc
```

Nếu không sử dụng tùy chọn `-n`, `sed` sẽ in mọi thứ từ tệp ngoại trừ những gì `d` đã hướng dẫn `sed` xóa khỏi đầu ra của nó.

Cách sử dụng phổ biến của `sed` là tìm và thay thế văn bản trong một tệp. Giả sử chúng ta muốn thay đổi mọi lần xuất hiện của `cat` thành `dog`. Ta có thể sử dụng `sed` để làm điều này bằng cách cung cấp tùy chọn `s` để hoán đổi từng trường hợp của thuật ngữ đầu tiên (`cat`) cho thuật ngữ thứ hai (`dog`):

```
$ sed s/cat/dog/ < ftu.txt
bzdog
dog
cut
head
less
md5sum
nl
od
paste
sed
sha256sum
sha512sum
sort
```

```
split
tail
tr
uniq
wc
xzdog
zdog
```

Thay vì sử dụng toán tử chuyển hướng (<) để truyền tệp `ftu.txt` vào lệnh `sed`, chúng ta chỉ cần lệnh `sed` hoạt động trực tiếp trên tệp. Chúng ta sẽ thử điều này ngay sau đây, đồng thời tạo một bản sao lưu của tệp gốc:

```
$ sed -i.backup s/cat/dog/ ftu.txt
$ ls ftu*
ftu.txt  ftu.txt.backup
```

Tùy chọn `-i` sẽ thực hiện thao tác `sed` tại chỗ trên tệp *gốc*. Nếu không sử dụng `.backup` sau tham số `-i`, điều đó có nghĩa là ta đã vừa viết lại chính tệp *gốc* của mình. Bất cứ nội dung nào được sử dụng làm văn bản sau tham số `-i` đều sẽ trở thành tên của tệp gốc được lưu trước khi các sửa đổi mà người dùng đã yêu cầu `sed` thực hiện xảy ra.

Đảm bảo tính toàn vẹn của Dữ liệu

Chúng ta đã chứng minh được việc thao tác với các tệp trong Linux là rất dễ dàng. Đôi khi, ta có thể sẽ muốn phân phối tệp cho người khác và cần chắc chắn rằng người nhận sẽ nhận được bản sao thực của tệp gốc. Một trường hợp sử dụng rất phổ biến của kỹ thuật này là khi các máy chủ phân phối Linux lưu trữ các hình ảnh CD hoặc DVD có thể tải xuống của phần mềm cùng với các tệp chứa các giá trị tổng kiểm tra được tính toán của các hình ảnh đĩa đó. Dưới đây là một kiểu ví dụ liệt kê từ máy bản sao tải xuống Debian (Debian download mirror):

```
_____________________________________________________________________
[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
```

Trong danh sách trên, các tệp hình ảnh của bộ cài đặt Debian được đi kèm theo cùng các tệp văn bản có chứa tổng kiểm tra của các tệp từ các thuật toán khác nhau (MD5, SHA1, SHA256 và SHA512).

NOTE

Tổng kiểm tra là một giá trị bắt nguồn từ một phép tính toán học đối với một tệp dựa trên một hàm băm mật mã. Có nhiều loại hàm băm mật mã khác nhau với độ mạnh khác nhau. Kỳ thi sẽ yêu cầu bạn làm quen với việc sử dụng `md5sum`, `sha256sum` và `sha512sum`.

Sau khi đã tải xuống một tệp (ví dụ: hình ảnh `debian-10.1.0-amd64-netinst.iso`), chúng ta sẽ so sánh tổng kiểm tra của tệp được tải xuống với giá trị tổng kiểm tra được cung cấp.

Dưới đây là một ví dụ để minh họa. Chúng ta sẽ tính giá trị SHA256 của tệp `ftu.txt` bằng cách sử dụng lệnh `sha256sum`:

```
$ sha256sum ftu.txt
345452304fc26999a715652543c352e5fc7ee0c1b9deac6f57542ec91daf261c  ftu.txt
```

Chuỗi ký tự dài trước tên tệp là giá trị tổng kiểm tra SHA256 của tệp văn bản này. Hãy cùng tạo một tệp chứa giá trị đó để ta có thể sử dụng nó và xác minh tính toàn vẹn của tệp văn bản gốc. Chúng ta có thể làm điều này với cùng một lệnh `sha256sum` và chuyển hướng đầu ra sang một tệp:

```
$ sha256sum ftu.txt > sha256.txt
```

Bây giờ, để xác minh tệp `ftu.txt`, ta chỉ cần sử dụng cùng một lệnh và cung cấp tên tệp chứa giá trị tổng kiểm tra cùng với khoá chuyển `-c`:

```
$ sha256sum -c sha256.txt
ftu.txt: OK
```

Giá trị chứa trong tệp khớp với tổng kiểm tra SHA256 đã tính cho tệp `ftu.txt`, đúng như chúng ta mong đợi. Tuy nhiên, nếu tệp gốc đã bị sửa đổi (chẳng hạn như một vài byte bị mất trong quá trình tải xuống tệp hoặc ai đó đã cố tình can thiệp vào tệp đó) thì việc kiểm tra giá trị sẽ không thành công. Trong những trường hợp như vậy, chúng ta sẽ biết rằng tệp của mình đã bị lỗi hoặc bị

hỏng và sẽ không thể tin tưởng vào tính toàn vẹn của nội dung. Để chứng minh điều này, hãy cùng thêm một phần văn bản vào cuối tệp:

```
$ echo "new entry" >> ftu.txt
```

Bây giờ, chúng ta sẽ thử xác minh tính toàn vẹn của tệp:

```
$ sha256sum -c sha256.txt
ftu.txt: FAILED
sha256sum: WARNING: 1 computed checksum did NOT match
```

Và chúng ta thấy được rằng tổng kiểm tra không khớp với những gì được mong đợi đối với tệp. Do đó, chúng ta không thể tin tưởng vào tính toàn vẹn của tệp này. Ta có thể thử tải xuống một bản sao mới của tệp, báo cáo lỗi tổng kiểm tra cho người gửi tệp hoặc báo cáo cho nhóm bảo mật trung tâm dữ liệu, tùy thuộc vào mức độ quan trọng của tệp.

Phân tích kỹ hơn về Tệp

Lệnh kết xuất bát phân (`od`) thường được sử dụng để gỡ lỗi các ứng dụng và các tệp khác nhau. Bản thân lệnh `od` sẽ chỉ liệt kê nội dung của tệp ở định dạng bát phân. Chúng ta có thể sử dụng tệp `ftu.txt` trước đó để thực hành với lệnh này:

```
$ 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
```

Cột đầu tiên của đầu ra là *phần bù byte* cho mỗi dòng đầu ra. Vì `od` in ra thông tin ở định dạng bát phân theo mặc định nên mỗi dòng đều sẽ bắt đầu bằng phần bù byte gồm tám bit, theo sau là tám cột, mỗi cột đều chứa giá trị bát phân của dữ liệu trong cột đó.

TIP Hãy nhớ rằng một *byte* có độ dài 8 bit.

Nếu cần xem nội dung của tệp ở định dạng thập lục phân, hãy sử dụng tùy chọn `-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
```

Giờ đây, mỗi cột trong số tám cột sau cột phần bù byte đều được biểu thị bằng các giá trị thập lục phân tương đương của chúng.

Một cách sử dụng hữu ích của lệnh `od` là để gỡ lỗi các tệp lệnh. Ví dụ: lệnh `od` có thể hiển thị các ký tự thường không được nhìn thấy tồn tại trong một tệp, chẳng hạn như các mục nhập *xuống dòng*. Chúng ta có thể làm điều này với tùy chọn `-c` để các mục nhập trong cột này sẽ được hiển thị dưới dạng ký tự tương đương của chúng thay vì hiển thị ký hiệu số cho mỗi byte:

```
$ 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
```

Tất cả các mục nhập xuống dòng trong tệp đều được biểu thị bằng các ký tự `\n` ẩn. Nếu chỉ muốn xem tất cả các ký tự trong một tệp và không cần xem thông tin phần bù byte, cột phần bù byte có thể được xóa khỏi đầu ra như sau:

```
$ 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
```

Bài tập Hướng dẫn

1. Ai đó vừa tặng một chiếc máy tính xách tay cho trường học của bạn và bạn muốn cài đặt Linux trên đó. Không có hướng dẫn sử dụng và bạn buộc phải khởi động nó từ ổ USB mà không có bất kỳ đồ họa nào. Bạn có một thiết bị đầu cuối tương tác vỏ và đối với mỗi bộ xử lý bạn có sẽ có một dòng dành cho nó trong tệp `/proc/cpuinfo`:

```
processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(lines skipped)

processor   : 1
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(more lines skipped)
```

   - Hãy sử dụng các lệnh `grep` và `wc` để hiển thị số lượng bộ xử lý bạn có.

   - Hãy làm điều tương tự với `sed` thay vì `grep`.

2. Hãy khám phá tệp `/etc/passwd` cục bộ của bạn bằng các lệnh `grep`, `sed`, `head` và `tail` cho mỗi tác vụ bên dưới:

   - Những người dùng nào có quyền truy cập vào vỏ Bash?

   - Hệ thống của bạn có nhiều người dùng tồn tại để xử lý các chương trình cụ thể hoặc cho các mục đích quản trị. Họ không có quyền truy cập vào vỏ. Có bao nhiêu người dùng trong số đó tồn tại trong hệ thống?

   - Có bao nhiêu người dùng và nhóm tồn tại trong hệ thống (hãy nhớ: chỉ sử dụng tệp `/etc/passwd`)?

- Hãy chỉ liệt kê dòng đầu tiên, dòng cuối cùng và dòng thứ mười của tệp `/etc/passwd`.

3. Hãy xem ví dụ về tệp `/etc/passwd` này. Hãy sao chép các dòng bên dưới vào tệp cục bộ có tên `mypasswd` cho bài tập này.

```
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
```

- Hãy liệt kê tất cả những người dùng trong nhóm `1000` (sử dụng `sed` để chỉ chọn trường thích hợp) từ tệp `mypasswd` của bạn.

- Hãy chỉ liệt kê tên đầy đủ của tất cả những người dùng trong nhóm này (sử dụng `sed` và `cut`).

Bài tập Mở rộng

Một lần nữa, bằng cách sử dụng tệp `mypasswd` từ các bài tập trước, hãy nghĩ ra một lệnh Bash để chọn ra duy nhất một cá nhân từ Main Office giành chiến thắng trong cuộc thi xổ số. Hãy sử dụng lệnh `sed` để chỉ in ra các dòng dành cho Main Office; sau đó, hãy sử dụng chuỗi lệnh `cut` để truy xuất tên của mỗi người dùng từ các dòng này. Tiếp theo, hãy sắp xếp ngẫu nhiên các tên này và chỉ in ra tên đứng đầu trong danh sách.

+

1. Có bao nhiêu người làm việc trong lĩnh vực Tài chính, Kỹ thuật và Bán hàng? (Cân nhắc khám phá lệnh `uniq`.)

2. Bạn muốn chuẩn bị một tệp CSV (các giá trị được phân tách bằng dấu phẩy) để có thể dễ dàng nhập tệp `names.csv` vào LibreOffice từ tệp `mypasswd` trong ví dụ trước. Nội dung tệp sẽ có định dạng sau:

   ```
   First Name,Last Name,Position
   Carol,Smith,Finance
   ...
   John,Chapel,Sales
   ```

   Mẹo: Sử dụng các lệnh `sed`, `cut` và `paste` để đạt được kết quả mong muốn. Hãy lưu ý rằng dấu phẩy (`,`) sẽ là dấu phân cách cho tệp này.

3. Giả sử rằng bảng tính `names.csv` được tạo trong bài tập trước là một tệp quan trọng và chúng ta muốn đảm bảo rằng không ai có thể can thiệp vào bảng tính đó kể từ thời điểm chúng ta gửi tệp và thời điểm người nhận nhận được tệp. Làm cách nào để chúng ta có thể đảm bảo tính toàn vẹn của tệp này bằng cách sử dụng `md5sum`?

4. Bạn đã tự hứa với mình sẽ đọc một cuốn sách kinh điển, mỗi ngày 100 dòng và bạn quyết định bắt đầu với *Mariner and Mystic* của Herman Melville. Hãy nghĩ ra một lệnh bằng cách sử dụng `split` để tách cuốn sách này thành các phần, mỗi phần dài 100 dòng. Để có được cuốn sách ở định dạng văn bản thuần túy, hãy tìm nó tại https://www.gutenberg.org.

5. Bằng cách sử dụng `ls -l` trên thư mục `/etc`, bạn sẽ nhận được loại danh sách nào? Bạn sẽ chỉ

hiển thị tên tệp bằng cách nào khi sử dụng lệnh `cut` trên đầu ra của lệnh `ls` đã cho? Còn tên tệp và chủ sở hữu của các tệp thì sao? Cùng với các lệnh `ls -l` và `cut`, hãy sử dụng lệnh `tr` để *nén* nhiều khoảng trắng thành một khoảng trắng để hỗ trợ định dạng đầu ra bằng lệnh `cut`.

6. Bài tập này giả định rằng bạn đang sử dụng một máy thật (không phải máy ảo). Bạn cũng phải có một chiếc USB. Hãy xem lại các trang hướng dẫn cho lệnh `tail` và tìm hiểu cách theo dõi tệp khi văn bản được thêm vào tệp. Trong khi theo dõi đầu ra của lệnh `tail` trên tệp `/var/log/syslog`, hãy cắm USB. Hãy viết ra lệnh đầy đủ mà bạn sẽ sử dụng để lấy Tên Sản phẩm, Nhà sản xuất và Tổng dung lượng bộ nhớ của USB.

Tóm tắt

Xử lý các luồng văn bản có tầm quan trọng lớn trong việc quản trị bất kỳ hệ thống nào trong Linux. Các luồng văn bản có thể được xử lý bằng tệp lệnh để tự động hóa các tác vụ hàng ngày hoặc tìm thông tin gỡ lỗi có liên quan trong tệp nhật ký. Dưới đây là một bản tóm tắt ngắn về các lệnh đã được nhắc đến trong bài học này:

`cat`

Được sử dụng để kết hợp hoặc đọc các tệp văn bản thuần túy.

`bzcat`

Cho phép xử lý hoặc đọc các tệp được nén bằng phương pháp `bzip2`.

`xzcat`

Cho phép xử lý hoặc đọc các tệp được nén bằng phương pháp `xz`.

`zcat`

Cho phép xử lý hoặc đọc các tệp được nén bằng phương pháp `gzip`.

`less`

Phân trang nội dung của tệp và cho phép chức năng điều hướng và tìm kiếm.

`head`

Hiển thị 10 dòng đầu tiên của tệp. Với việc sử dụng khóa chuyển `-n`, có thể hiển thị ít hoặc nhiều dòng hơn.

`tail`

Hiển thị 10 dòng cuối cùng của tệp. Với việc sử dụng khóa chuyển `-n`, nó có thể hiển thị ít hoặc nhiều dòng hơn. Tùy chọn `-f` được sử dụng để theo dõi đầu ra của tệp văn bản có dữ liệu mới đang được ghi vào đó.

`wc`

Viết tắt của “word count” (số từ), nhưng tùy thuộc vào thông số bạn sử dụng, nó sẽ đếm ký tự, từ và dòng.

`sort`

Được sử dụng để sắp xếp đầu ra của danh sách theo thứ tự bảng chữ cái, đảo ngược thứ tự bảng chữ cái hoặc theo thứ tự ngẫu nhiên.

`uniq`

Được sử dụng để liệt kê (và đếm) các chuỗi khớp.

`od`

Lệnh “octal dump” được sử dụng để hiển thị tệp nhị phân theo ký hiệu bát phân, thập phân hoặc thập lục phân.

`nl`

Lệnh “number line” sẽ hiển thị số dòng trong một tệp cũng như tạo lại một tệp với mỗi dòng được thêm vào trước bởi số dòng của nó.

`sed`

Trình chỉnh sửa luồng có thể được sử dụng để tìm các lần xuất hiện khớp của chuỗi bằng cách sử dụng Biểu thức chính quy cũng như chỉnh sửa tệp bằng cách sử dụng các mẫu được xác định trước.

`tr`

Lệnh dịch có thể thay thế các ký tự, đồng thời loại bỏ và nén các ký tự lặp lại.

`cut`

Lệnh này có thể in các cột của tệp văn bản dưới dạng các trường dựa trên dấu phân cách ký tự của tệp.

`paste`

Nối các tệp trong các cột dựa trên việc sử dụng dấu tách trường.

`split`

Lệnh này có thể chia các tệp lớn thành các tệp nhỏ hơn tùy thuộc vào tiêu chí được đặt bởi các tùy chọn của lệnh.

`md5sum`

Được sử dụng để tính toán giá trị băm MD5 của tệp. Cũng được sử dụng để xác minh tệp dựa trên giá trị băm hiện có để đảm bảo tính toàn vẹn của tệp.

`sha256sum`

Được sử dụng để tính toán giá trị băm SHA256 của tệp. Cũng được sử dụng để xác minh tệp dựa trên giá trị băm hiện có để đảm bảo tính toàn vẹn của tệp.

`sha512sum`

Được sử dụng để tính toán giá trị băm SHA512 của tệp. Cũng được sử dụng để xác minh tệp dựa

trên giá trị băm hiện có để đảm bảo tính toàn vẹn của tệp.

Đáp án Bài tập Hướng dẫn

1. Ai đó vừa tặng một chiếc máy tính xách tay cho trường học của bạn và bạn muốn cài đặt Linux trên đó. Không có hướng dẫn sử dụng và bạn buộc phải khởi động nó từ ổ USB mà không có bất kỳ đồ họa nào. Bạn có một thiết bị đầu cuối tương tác vỏ và đối với mỗi bộ xử lý bạn có sẽ có một dòng dành cho nó trong tệp `/proc/cpuinfo`:

```
processor   : 0
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(lines skipped)

processor   : 1
vendor_id   : GenuineIntel
cpu family  : 6
model       : 158

(more lines skipped)
```

   - Hãy sử dụng các lệnh `grep` và `wc` để hiển thị số lượng bộ xử lý bạn có.

     Here are two options:

```
$ cat /proc/cpuinfo | grep processor | wc -l
$ grep processor /proc/cpuinfo | wc -l
```

     Bạn đã biết có một số cách có thể làm điều tương tự, vậy khi nào bạn nên sử dụng chúng? Điều này phụ thuộc vào một số yếu tố, hai yếu tố quan trọng nhất là hiệu suất và khả năng đọc. Hầu như bạn sẽ sử dụng các lệnh vỏ bên trong các tệp lệnh vỏ để tự động hóa các tác vụ của mình. Các tệp lệnh càng lớn và phức tạp thì bạn càng phải lo lắng về việc duy trì tốc độ của chúng.

   - Hãy làm điều tương tự với `sed` thay vì `grep`.

     Bây giờ, thay vì `grep`, chúng ta sẽ thử điều này với `sed`:

```
$ sed -n /processor/p /proc/cpuinfo | wc -l
```

Ở đây, chúng ta đã sử dụng `sed` với tham số `-n`. Vì vậy, `sed` sẽ không in bất kỳ thứ gì ngoại trừ những gì khớp với biểu thức `processor` theo hướng dẫn của lệnh `p`. Như chúng ta đã làm trong các giải pháp `grep`, `wc -l` sẽ đếm số dòng - cũng chính là số lượng bộ xử lý mà chúng ta có.

Hãy xem ví dụ tiếp theo này:

```
$ sed -n /processor/p /proc/cpuinfo | sed -n '$='
```

Chuỗi lệnh này cung cấp các kết quả giống hệt với ví dụ trước, trong đó, đầu ra của `sed` được chuyển thành lệnh `wc`. Sự khác biệt ở đây là thay vì sử dụng `wc -l` để đếm số dòng, `sed` lại được gọi để cung cấp chức năng tương đương. Một lần nữa, chúng ta đang chặn đầu ra của `sed` với tùy chọn `-n`, ngoại trừ biểu thức mà chúng ta đang gọi một cách rõ ràng, tức `'$='`. Biểu thức này yêu cầu `sed` khớp với dòng cuối cùng (`$`) và sau đó sẽ in số dòng đó (`=`).

2. Hãy khám phá tệp `/etc/passwd` cục bộ của bạn bằng các lệnh `grep`, `sed`, `head` và `tail` cho mỗi tác vụ bên dưới:
   - Những người dùng nào có quyền truy cập vào vỏ Bash?

     ```
     $ grep ":/bin/bash$" /etc/passwd
     ```

     Chúng ta có thể cải thiện câu trả lời này bằng cách chỉ hiển thị tên của người dùng sử dụng vỏ Bash.

     ```
     $ grep ":/bin/bash$" /etc/passwd | cut -d: -f1
     ```

     Tên người dùng là trường đầu tiên (tham số `-f1` của lệnh `cut`) và tệp `/etc/passwd` sử dụng `:` làm dấu phân cách (tham số `-d:` của lệnh `cut`), chúng ta chỉ dẫn đầu ra của lệnh `grep` sang lệnh `cut` thích hợp.

   - Hệ thống của bạn có nhiều người dùng tồn tại để xử lý các chương trình cụ thể hoặc cho các mục đích quản trị. Họ không có quyền truy cập vào vỏ. Có bao nhiêu người dùng trong số đó tồn tại trong hệ thống?

     Cách dễ nhất để tìm ra điều này là in ra các dòng cho các tài khoản không sử dụng vỏ Bash:

     ```
     $ grep -v ":/bin/bash$" /etc/passwd | wc -l
     ```

- Có bao nhiêu người dùng và nhóm tồn tại trong hệ thống (hãy nhớ: chỉ sử dụng tệp `/etc/passwd`)?

  Trường đầu tiên của bất kỳ dòng cụ thể nào trong tệp `/etc/passwd` sẽ là tên người dùng, trường thứ hai thường là `x` cho biết mật khẩu người dùng không được lưu trữ ở đây (nó sẽ được mã hóa trong tệp `/etc/shadow`), thứ ba là id người dùng (UID) và thứ tư là id nhóm (GID). Vì vậy, mã sau sẽ cho chúng ta biết về số lượng người dùng:

  ```
  $ cut -d: -f3 /etc/passwd | wc -l
  ```

  Tuy nhiên, có những trường hợp mà trong đó, bạn đã thiết lập các siêu người dùng khác nhau hoặc các loại người dùng đặc biệt khác chia sẻ cùng một UID (id người dùng). Vì vậy, để đảm bảo an toàn, chúng ta sẽ dẫn kết quả của lệnh `cut` sang lệnh `sort` và sau đó đếm số dòng.

  ```
  $ cut -d: -f3 /etc/passwd | sort -u | wc -l
  ```

  Đối với số lượng nhóm:

  ```
  $ cut -d: -f4 /etc/passwd | sort -u | wc -l
  ```

- Hãy chỉ liệt kê tên đầy đủ của tất cả những người dùng trong nhóm này (sử dụng `sed` và `cut`).

  This will do:

  ```
  $ sed -n -e '1'p -e '10'p -e '$'p /etc/passwd
  ```

  Hãy nhớ rằng tham số `-n` sẽ yêu cầu `sed` không in bất cứ thứ gì ngoài những gì được chỉ định bởi lệnh `p`. Ký hiệu đô la (`$`) được sử dụng ở đây là biểu thức chính quy (có nghĩa là dòng cuối cùng của tệp).

3. Hãy xem ví dụ về tệp `/etc/passwd` này. Hãy sao chép các dòng bên dưới vào tệp cục bộ có tên `mypasswd` cho bài tập này.

   ```
   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
```

- Hãy liệt kê tất cả những người dùng trong nhóm `1000` (sử dụng `sed` để chỉ chọn trường thích hợp) từ tệp `mypasswd` của bạn.

  GID là trường thứ tư trong tệp `/etc/passwd`. Có thể bạn sẽ muốn thử mã này:

  ```
  $ sed -n /1000/p mypasswd
  ```

  Trong trường hợp này, bạn cũng sẽ nhận được dòng sau:

  ```
  carol:x:1000:2000:Carol Smith,Finance,,,Main Office:/home/carol:/bin/bash
  ```

  Bạn biết điều này không chính xác vì Carol Smith là thành viên của GID 2000 và nó khớp là do UID. Tuy nhiên, bạn có thể đã nhận thấy rằng sau GID, trường tiếp theo đã bắt đầu bằng ký tự chữ hoa. Chúng ta có thể sử dụng một biểu thức chính quy để giải quyết vấn đề này.

  ```
  $ sed -n /:1000:[A-Z]/p mypasswd
  ```

  Biểu thức `[A-Z]` sẽ khớp với bất kỳ ký tự chữ hoa nào. Bạn sẽ tìm hiểu thêm về điều này trong bài học tương ứng.

- Hãy chỉ liệt kê tên đầy đủ của tất cả những người dùng trong nhóm này (sử dụng `sed` và `cut`).

  Sử dụng kỹ thuật tương tự mà bạn đã dùng để giải phần đầu tiên của bài tập này và dẫn nó thành lệnh `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
```

Chưa xong đâu! Hãy lưu ý cách các trường bên trong kết quả của bạn có thể được phân tách bằng dấu `,`. Vì vậy, chúng ta sẽ dẫn đầu ra sang một lệnh `cut` khác và sử dụng `,` làm dấu phân cách.

```
$ 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
```

Đáp án Bài tập Mở rộng

Một lần nữa, bằng cách sử dụng tệp `mypasswd` từ các bài tập trước, hãy nghĩ ra một lệnh Bash để chọn ra duy nhất một cá nhân từ Main Office giành chiến thắng trong cuộc thi xổ số. Hãy sử dụng lệnh `sed` để chỉ in ra các dòng dành cho Main Office; sau đó, hãy sử dụng chuỗi lệnh `cut` để truy xuất tên của mỗi người dùng từ các dòng này. Tiếp theo, hãy sắp xếp ngẫu nhiên các tên này và chỉ in ra tên đứng đầu trong danh sách.

+ Trước tiên, hãy khám phá cách tham số `-R` điều khiển đầu ra của lệnh `sort` (sắp xếp). Hãy lặp lại lệnh này một vài lần trên máy của bạn (lưu ý rằng bạn cần đặt 'Main Office' trong dấu trích dẫn đơn để `sed` xử lý nó dưới dạng một chuỗi):

+

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1 | sort -R
```

+ Giải pháp cho vấn đề này sẽ là:

+

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1 | sort -R | head -1
```

1. Có bao nhiêu người làm việc trong lĩnh vực Tài chính, Kỹ thuật và Bán hàng? (Cân nhắc khám phá lệnh `uniq`.)

   Hãy tiếp tục dựa trên những gì bạn đã học được từ các bài tập trước. Hãy thử như sau:

   ```
   $ sed -n /'Main Office'/p mypasswd
   $ sed -n /'Main Office'/p mypasswd | cut -d, -f2
   ```

   Lưu ý rằng bây giờ chúng ta không quan tâm đến `:` như một dấu phân cách. Ta chỉ cần trường thứ hai khi chia các dòng theo các ký tự `,`.

   ```
   $ sed -n /'Main Office'/p mypasswd | cut -d, -f2 | uniq -c
      4 Finance
      1 Engineering
      2 Sales
   ```

   Lệnh `uniq` sẽ chỉ xuất ra các dòng duy nhất (không phải các dòng lặp lại) và tham số `-c` cho

`uniq` đếm số lần xuất hiện của các dòng ngang nhau. Có một lưu ý ở đây: `uniq` sẽ chỉ xem xét các dòng liền kề. Nếu không phải, bạn sẽ phải sử dụng lệnh `sort`.

2. Bạn muốn chuẩn bị một tệp CSV (các giá trị được phân tách bằng dấu phẩy) để có thể dễ dàng nhập tệp `names.csv` vào LibreOffice từ tệp `mypasswd` trong ví dụ trước. Nội dung tệp sẽ có định dạng sau:

```
First Name,Last Name,Position
Carol,Smith,Finance
...
John,Chapel,Sales
```

Mẹo: Sử dụng các lệnh `sed`, `cut` và `paste` để đạt được kết quả mong muốn. Hãy lưu ý rằng dấu phẩy (`,`) sẽ là dấu phân cách cho tệp này.

Bắt đầu với các lệnh `sed` và `cut` dựa trên những gì chúng ta đã học được từ các bài tập trước:

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d" " -f1 > firstname
```

Bây giờ, chúng ta đã có tệp `firstname` với tên của các nhân viên.

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d" " -f2 | cut -d, -f1 > lastname
```

Bây giờ, chúng ta đã có tệp `lastname` chứa họ của từng nhân viên.

Tiếp theo, chúng ta sẽ xác định xem mỗi nhân viên làm việc trong bộ phận nào:

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f2 > department
```

Trước khi tiến tới giải pháp cuối cùng, hãy thử các lệnh sau để xem chúng tạo ra loại đầu ra nào:

```
$ cat firstname lastname department
$ paste firstname lastname department
```

Và sau đó là giải pháp cuối cùng:

```
$ paste firstname lastname department | tr '\t' ,
```

```
$ paste firstname lastname department | tr '\t' , > names.csv
```

Ở đây, chúng ta đã sử dụng lệnh `tr` để *dịch* `\t` (dấu tách tab) bằng dấu `,`. Lệnh `tr` khá hữu ích khi chúng ta cần trao đổi các ký tự với nhau. Hãy chắc chắn là bạn có xem lại các trang hướng dẫn cho cả `tr` và `paste`. Ví dụ: chúng ta có thể sử dụng tùy chọn `-d` cho dấu phân cách để làm cho lệnh trước đó bớt phức tạp hơn:

```
$ paste -d, firstname lastname department
```

Chúng ta đã sử dụng lệnh `paste` ở đây để giúp bạn làm quen với nó. Tuy nhiên, chúng ta có thể dễ dàng thực hiện tất cả các tác vụ trong một chuỗi lệnh duy nhất:

```
$ sed -n /'Main Office'/p mypasswd | cut -d: -f5 | cut -d, -f1,2 | tr ' ' , > names.csv
```

3. Giả sử rằng bảng tính `names.csv` được tạo trong bài tập trước là một tệp quan trọng và chúng ta muốn đảm bảo rằng không ai có thể can thiệp vào bảng tính đó kể từ thời điểm ta gửi tệp cho ai đó và thời điểm người nhận nhận được tệp. Làm cách nào để chúng ta có thể đảm bảo tính toàn vẹn của tệp này bằng cách sử dụng `md5sum`?

   Nếu xem các trang hướng dẫn cho `md5sum`, `sha256sum` và `sha512sum`, bạn sẽ thấy tất cả đều bắt đầu bằng văn bản sau:

   “compute and check XXX message digest”

   Trong đó, “XXX” là thuật toán sẽ được sử dụng để tạo *thông báo tóm tắt* này.

   Chúng ta sẽ sử dụng `md5sum` làm ví dụ và bạn có thể thử với các lệnh khác sau này.

   ```
   $ md5sum names.csv
   61f0251fcab61d9575b1d0cbf0195e25 names.csv
   ```

   Ví dụ: bây giờ, bạn có thể cung cấp tệp thông qua dịch vụ ftp an toàn và gửi *thông báo tóm tắt* được tạo bằng cách sử dụng một phương tiện liên lạc an toàn khác. Nếu tệp đã được sửa đổi dù chỉ một chút thì *thông báo tóm tắt* cũng sẽ hoàn toàn khác. Để chứng minh điều này, hãy chỉnh sửa `names.csv` và đổi Jones thành James như trong minh họa ở đây:

   ```
   $ sed -i.backup s/Jones/James/ names.csv
   $ md5sum names.csv
   f44a0d68cb480466099021bf6d6d2e65 names.csv
   ```

Bất cứ khi nào cung cấp tệp để tải xuống, bạn cũng nên phân phối *thông báo tóm tắt* tương ứng để những người tải xuống tệp của bạn có thể tạo *thông báo tóm tắt* mới và đối chiếu với bản gốc. Nếu duyệt qua https://kernel.org, bạn sẽ tìm thấy trang https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/sha256sums.asc nơi có thể lấy sha256sum cho tất cả các tệp có sẵn để tải xuống.

4. Bạn đã tự hứa với mình rằng bạn sẽ đọc một cuốn sách kinh điển, mỗi ngày 100 dòng và bạn quyết định bắt đầu với *Mariner and Mystic* của Herman Melville. Hãy nghĩ ra một lệnh bằng cách sử dụng `split` để tách cuốn sách này thành các phần, mỗi phần dài 100 dòng. Để có được cuốn sách ở định dạng văn bản thuần túy, hãy tìm nó tại https://www.gutenberg.org.

   Đầu tiên, chúng ta sẽ lấy toàn bộ cuốn sách từ trang Project Gutenberg nơi bạn có thể lấy cuốn sách này và những cuốn sách khác có sẵn trong phạm vi công cộng.

   ```
   $ wget https://www.gutenberg.org/files/50461/50461-0.txt
   ```

   Có thể bạn sẽ cần phải cài đặt `wget` nếu chưa có nó trong hệ thống (ngoài ra, bạn cũng có thể sử dụng `curl`). Hãy sử dụng `less` để xác minh sách:

   ```
   $ less 50461-0.txt
   ```

   Bây giờ, chúng ta sẽ chia cuốn sách thành từng phần, mỗi phần có 100 dòng:

   ```
   $ split -l 100 -d 50461-0.txt melville
   ```

   `50461-0.txt` là tệp mà chúng ta sẽ chia tách. `melville` sẽ là tiền tố cho các tệp được chia. `-l 100` chỉ định số dòng và tùy chọn `-d` sẽ yêu cầu `split` đánh số các tệp (sử dụng hậu tố được cung cấp). Bạn có thể sử dụng `nl` trên bất kỳ tệp nào được chia tách (có thể không phải trên tệp cuối cùng) và xác nhận mỗi tệp có 100 dòng.

5. Bằng cách sử dụng `ls -l` trên thư mục `/etc`, bạn sẽ nhận được loại danh sách nào? Bạn sẽ chỉ hiển thị tên tệp bằng cách nào khi sử dụng lệnh `cut` trên đầu ra của lệnh `ls` đã cho? Còn tên tệp và chủ sở hữu của các tệp thì sao? Cùng với các lệnh `ls -l` và `cut`, hãy sử dụng lệnh `tr` để *nén* nhiều khoảng trắng thành một khoảng trắng để hỗ trợ định dạng đầu ra bằng lệnh `cut`.

   Bản thân lệnh `ls` sẽ chỉ cung cấp cho bạn tên của các tệp. Tuy nhiên, chúng ta có thể chuẩn bị đầu ra của `ls -l` (danh sách dài) để trích xuất thông tin cụ thể hơn.

   ```
   $ 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
```

Tham số `-s` sẽ hướng dẫn `tr` thu nhỏ các khoảng trắng lặp lại thành một bản thể duy nhất chỉ có một khoảng trắng. Lệnh `tr` sẽ hoạt động đối với mọi loại ký tự lặp lại mà bạn chỉ định. Sau đó, chúng ta sẽ thay thế khoảng trắng bằng dấu phẩy `,`. Vì không cần thay thế các khoảng trắng trong ví dụ nên chúng ta sẽ chỉ bỏ qua `,`.

```
$ 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
```

Nếu chỉ muốn tên tệp thì chúng ta chỉ cần hiển thị trường thứ chín:

```
$ ls -l /etc | tr -s ' ' | cut -d" " -f9
```

Đối với tên tệp và chủ sở hữu tệp, chúng ta sẽ cần trường thứ chín và thứ ba:

```
$ ls -l /etc | tr -s ' ' | cut -d" " -f9,3
```

Nếu chúng ta chỉ cần tên thư mục và chủ sở hữu của nó thì sao?

```
$ ls -l /etc | grep ^d | tr -s ' ' | cut -d" " -f9,3
```

6. Bài tập này giả định rằng bạn đang sử dụng một máy thật (không phải máy ảo). Bạn cũng phải có một chiếc USB. Hãy xem lại các trang hướng dẫn cho lệnh `tail` và tìm hiểu cách theo dõi tệp

khi văn bản được thêm vào tệp. Trong khi theo dõi đầu ra của lệnh `tail` trên tệp `/var/log/syslog`, hãy cắm USB. Hãy viết ra lệnh đầy đủ mà bạn sẽ sử dụng để lấy Tên Sản phẩm, Nhà sản xuất và Tổng dung lượng bộ nhớ của USB.

```
$ 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)
```

Tất nhiên đây chỉ là một ví dụ và kết quả sẽ khác biệt tùy thuộc vào từng nhà sản xuất USB. Bây giờ, hãy lưu ý rằng chúng ta đã sử dụng tham số `-i` với lệnh `grep` vì ta không chắc liệu các chuỗi mình đang tìm kiếm là chữ hoa hay chữ thường. Chúng ta cũng đã sử dụng `|` như một Hàm OR logic nên thực tế chúng ta đang tìm kiếm các dòng có chứa `product` HOẶC `blocks` HOẶC `manufacturer`.

103.3 Thực hiện Quản lý Tệp cơ bản

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 103.3

Khối lượng

4

Các lĩnh vực kiến thức chính

- Sao chép, di chuyển và xóa các tệp và thư mục riêng lẻ.
- Sao chép đệ quy nhiều tệp và thư mục.
- Loại bỏ các tệp và thư mục theo cách đệ quy.
- Sử dụng các thông số kỹ thuật ký tự đại diện đơn giản và nâng cao trong các lệnh.
- Sử dụng lệnh find để định vị và hành động trên các tệp dựa trên loại, kích thước hoặc thời gian.
- Cách sử dụng tar, cpio và dd.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

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

- `tar`
- `cpio`
- `dd`
- `file`
- `gzip`
- `gunzip`
- `bzip2`
- `bunzip2`
- file globbing

Linux Professional Institute

103.3 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.3 Thực hiện Quản lý Tệp cơ bản
Bài:	1 trên 2

Giới thiệu

Vì mọi thứ trong Linux đều được coi là một tệp nên việc học cách thao tác với chúng là một điều rất quan trọng. Trong bài học này, chúng ta sẽ thảo luận về các thao tác cơ bản trên tệp.

Nhìn chung, với tư cách là một người dùng Linux, chúng ta sẽ phải điều hướng trong hệ thống tệp, sao chép tệp từ vị trí này sang vị trí khác và xóa tệp. Bài học cũng sẽ đề cập đến các lệnh liên quan đến việc quản lý tệp.

Tệp là một thực thể lưu trữ dữ liệu và chương trình. Nó bao gồm nội dung và các siêu dữ liệu (kích thước tệp, chủ sở hữu, ngày tạo, quyền). Các tệp được tổ chức trong các thư mục, mỗi thư mục đều là một tệp lưu trữ các tệp khác.

Các loại tệp bao gồm:

Tệp thông thường
: lưu trữ dữ liệu và các chương trình.

Thư mục

chứa các tệp khác.

Tệp đặc biệt

được sử dụng cho đầu vào và đầu ra.

Tất nhiên là cũng có cả các loại tệp khác nữa, nhưng chúng nằm ngoài phạm vi của bài học này. Sau đây, chúng ta sẽ cùng thảo luận cách xác định các loại tệp khác nhau này.

Thao tác với Tệp

Sử dụng `ls` để liệt kê Tệp

Lệnh `ls` là một trong những công cụ dòng lệnh quan trọng nhất mà chúng ta nên tìm hiểu để điều hướng trong hệ thống tệp.

Ở dạng cơ bản, `ls` sẽ *chỉ* liệt kê các tên tệp và thư mục:

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

Khi được sử dụng với `-l` (định dạng "danh sách dài"), nó sẽ hiển thị quyền của tệp hoặc thư mục, chủ sở hữu, kích thước, ngày, giờ và tên đã sửa đổi:

```
$ 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
```

Ký tự đầu tiên trong đầu ra cho biết loại tệp:

-
: đối với tệp thông thường.

d
: đối với thư mục.

c
: đối với tệp đặc biệt.

Để hiển thị kích thước tệp ở định dạng con người có thể đọc được, hãy thêm tùy chọn `-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
```

Để liệt kê tất cả các tệp bao gồm cả các tệp ẩn (những tệp bắt đầu bằng `.`), hãy sử dụng tùy chọn `-a`:

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

Hiện chúng ta đã có thể thấy được các tệp cấu hình như `.bash_history` được ẩn theo mặc định.

Nói chung, cú pháp lệnh `ls` được đưa ra bởi:

```
ls OPTIONS FILE
```

Trong đó, `OPTIONS` là bất kỳ tùy chọn nào được hiển thị trước đó (để xem tất cả các tùy chọn có thể, hãy chạy `man ls`) và `FILE` là tên chi tiết của tệp hoặc thư mục mà người muốn liệt kê.

NOTE Khi `FILE` không được chỉ định thì lệnh sẽ mặc định trỏ tới thư mục hiện tại.

Tạo, sao chép, di chuyển và xóa Tệp

Tạo tệp bằng lệnh `touch`

Lệnh `touch` là cách dễ nhất để tạo các tệp trống mới. Ta cũng có thể sử dụng nó để thay đổi dấu thời gian (nghĩa là thời gian sửa đổi) của các tệp và thư mục hiện có. Cú pháp để sử dụng lệnh `touch` là:

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

Nếu không có bất kỳ tùy chọn nào, `touch` sẽ tạo các tệp mới cho bất kỳ tên tệp nào được cung cấp dưới dạng đối số, với điều kiện là các tệp có tên như vậy chưa tồn tại. `touch` có thể tạo đồng thời bao nhiêu tệp cũng được:

```
$ touch file1 file2 file3
```

Thao tác này sẽ tạo ra ba tệp trống mới có tên `file1`, `file2` và `file3`.

Một số tùy chọn `touch` được thiết kế đặc biệt để cho phép người dùng thay đổi dấu thời gian cho các tệp. Ví dụ: tùy chọn `-a` chỉ thay đổi thời gian truy cập, trong khi tùy chọn `-m` sẽ chỉ thay đổi thời gian sửa đổi. Việc sử dụng cả hai tùy chọn cùng nhau sẽ thay đổi quyền truy cập và cả thời gian sửa đổi thành thời điểm hiện tại:

```
$ touch -am file3
```

Sao chép tệp bằng `cp`

Là người dùng Linux, bạn sẽ thường xuyên phải sao chép tệp từ vị trí này sang vị trí khác. Cho dù đó là di chuyển một tệp âm thanh từ thư mục này sang một thư mục khác hoặc sang một tệp hệ thống, hãy sử dụng `cp` cho tất cả các tác vụ sao chép:

```
$ cp file1 dir2
```

Lệnh này có thể được hiểu theo nghĩa đen là sao chép `file1` vào thư mục `dir2`. Kết quả là `file1` đã có mặt bên trong `dir2`. Để lệnh này được thực thi thành công, `file1` phải tồn tại trong thư mục hiện tại của người dùng. Nếu không, hệ thống sẽ báo lỗi `No such file or directory` (Không có tệp hoặc thư mục như vậy).

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

Trong trường hợp này, hãy lưu ý rằng đường dẫn đến `file1` sẽ rõ ràng hơn. Đường dẫn nguồn có thể được biểu thị dưới dạng đường dẫn *tương đối* hoặc *tuyệt đối*. Các đường dẫn tương đối sẽ được đưa ra trong tham chiếu đến một thư mục cụ thể, trong khi các đường dẫn tuyệt đối sẽ không được đưa ra cùng với bất kỳ một tham chiếu nào. Sau đây, chúng ta sẽ cùng làm rõ thêm khái niệm này.

Hiện tại, chúng ta chỉ cần quan sát lệnh này sao chép tệp `file1` vào thư mục `dir2`. Đường dẫn đến `file1` được cung cấp nhiều chi tiết hơn vì người dùng hiện không ở trong thư mục `dir1`.

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

Trong trường hợp thứ ba này, tệp `file2` nằm ở `/home/frank/Documents` đã được sao chép vào thư mục `/home/frank/Documents/Backup`. Đường dẫn nguồn được cung cấp ở đây là *tuyệt đối*. Trong hai ví dụ trên, đường dẫn nguồn là *tương đối*. Khi một đường dẫn bắt đầu bằng ký tự `/` thì đó là đường dẫn tuyệt đối, nếu không thì đó là đường dẫn tương đối.

Cú pháp chung cho `cp` là:

```
cp OPTIONS SOURCE DESTINATION
```

`SOURCE` là tệp để sao chép và `DESTINATION` là thư mục mà tệp sẽ được sao chép vào. `SOURCE` và `DESTINATION` có thể được chỉ định dưới dạng đường dẫn tuyệt đối hoặc tương đối.

Di chuyển Tệp bằng `mv`

Giống như lệnh `cp` dùng để sao chép, Linux cũng cung cấp một lệnh dùng để di chuyển và đổi tên tệp là `mv`.

Thao tác di chuyển cũng tương tự như thao tác cắt và dán mà người dùng thường thực hiện thông qua Giao diện Đồ hoạ Người dùng (GUI).

Nếu muốn di chuyển tệp vào một vị trí mới, hãy sử dụng `mv` theo cách sau:

```
mv FILENAME DESTINATION_DIRECTORY
```

Sau đây là một ví dụ:

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

Kết quả là `myfile.txt` đã được di chuyển đến đích `/home/frank/Documents`.

Để đổi tên một tệp, `mv` có thể được sử dụng theo cách sau:

```
$ mv old_file_name new_file_name
```

Thao tác này sẽ thay đổi tên của tệp từ `old_file_name` thành `new_file_name`.

Theo mặc định, `mv` sẽ không cần ta phải xác nhận (về mặt kỹ thuật là "sẽ không nhắc lệnh") nếu ta muốn ghi đè lên (đổi tên) một tệp hiện có. Tuy nhiên, chúng ta có thể cho phép hệ thống nhắc lệnh bằng cách sử dụng tùy chọn `-i`:

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

Lệnh này sẽ yêu cầu sự cho phép của người dùng trước khi ghi đè tệp `old_file_name` lên tệp `new_file_name`.

Ngược lại, nếu ta sử dụng tuỳ chọn `-f`:

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

thì lệnh sẽ bắt buộc ghi đè vào tệp mà không cần sự cho phép.

Xóa Tệp bằng `rm`

`rm` được sử dụng để xóa các tệp (ta có thể coi nó như một dạng viết tắt của từ "remove"). Hãy lưu ý rằng hành động xóa tệp thường sẽ không thể đảo ngược; do đó, chúng ta nên thận trọng khi sử dụng lệnh này.

```
$ rm file1
```

Lệnh này sẽ xóa tệp `file1`.

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

Lệnh này sẽ yêu cầu người dùng xác nhận trước khi xóa tệp `file1`. Hãy nhớ rằng chúng ta đã thấy tùy chọn `-i` khi sử dụng lệnh `mv` ở trên.

```
$ rm -f file1
```

Lệnh này sẽ bắt buộc xóa tệp `file1` mà không cần người dùng xác nhận.

Ta cũng có thể xóa nhiều tệp cùng một lúc:

```
$ rm file1 file2 file3
```

Trong ví dụ này, các tệp `file1`, `file2` và `file3` sẽ đồng thời bị xóa.

Cú pháp cho `rm` thường được đưa ra bởi:

```
rm OPTIONS FILE
```

Tạo và xóa Thư mục

Tạo Thư mục với `mkdir`

Việc tạo thư mục là rất quan trọng trong việc tổ chức các tệp và thư mục. Các tệp có thể được nhóm lại với nhau một cách hợp lý bằng cách giữ chúng ở bên trong một thư mục. Để tạo một thư mục, hãy sử dụng `mkdir`:

```
mkdir OPTIONS DIRECTORY_NAME
```

Trong đó, `DIRECTORY_NAME` là tên của thư mục sẽ được tạo. Lệnh có thể đồng thời tạo ra bao nhiêu thư mục cũng được.

```
$ mkdir dir1
```

sẽ tạo thư mục `dir1` trong thư mục hiện tại của người dùng.

```
$ mkdir dir1 dir2 dir3
```

Lệnh trên sẽ tạo ra ba thư mục `dir1`, `dir2` và `dir3` cùng một lúc.

Để tạo một thư mục cùng với các thư mục con của nó, hãy sử dụng tùy chọn `-p` ("parents"):

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

Lệnh này sẽ tạo cấu trúc thư mục `parents/children` (mẹ/con), tức là nó sẽ tạo các thư mục `parents` (mẹ) và `children` (con). `children` sẽ được đặt bên trong `parents`.

Xóa Thư mục bằng `rmdir`

`rmdir` sẽ xóa một thư mục nếu nó trống. Cú pháp của nó được đưa ra bởi:

```
rmdir OPTIONS DIRECTORY
```

trong đó, `DIRECTORY` có thể là một đối số hoặc một danh sách các đối số.

```
$ rmdir dir1
```

Lệnh này sẽ xóa tệp `file1`.

```
$ rmdir dir1 dir2
```

Lệnh này sẽ đồng thời xóa `dir1` và `dir2`.

Ta có thể xóa một thư mục cùng với thư mục con của nó:

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

Thao tác này sẽ xóa cấu trúc thư mục `parents/children`. Hãy lưu ý rằng nếu bất kỳ thư mục nào không trống, chúng sẽ không bị xóa.

Thao tác đệ quy của Tệp và Thư mục

Để thao tác với một thư mục và nội dung của nó, ta cần áp dụng quy tắc *đệ quy*. Đệ quy có nghĩa là thực hiện một hành động và lặp lại hành động đó xuống toàn bộ cây thư mục. Trong Linux, các tùy chọn `-r`, `-R` hoặc `--recursive` thường là chỉ các thao tác đệ quy.

Ngữ cảnh sau đây sẽ giúp chúng ta hiểu rõ hơn về đệ quy:

Người dùng liệt kê nội dung của thư mục `students`. Thư mục này chứa hai thư mục con là `level 1` và `level 2` và một tệp có tên `frank`. Bằng cách áp dụng đệ quy, lệnh `ls` sẽ liệt kê nội dung của `students`, tức là `level 1`, `level 2` và `frank` nhưng sẽ không kết thúc ở đó. Nó cũng sẽ nhập các thư mục con là `level 1` và `level 2` và liệt kê nội dung của chúng, v.v. xuống cây thư mục.

Liệt kê Đệ quy với `ls -R`

`ls -R` được sử dụng để liệt kê nội dung của một thư mục cùng với các thư mục con và tệp của nó.

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

mydirectory/newdirectory:
```

Trong danh sách trên, `mydirectory` cùng với toàn bộ nội dung của nó đã được liệt kê. Ta có thể quan sát `mydirectory` chứa thư mục con `newdirectory` và tệp `file1`. Vì thư mục `newdirectory` trống nên không có nội dung nào được hiển thị.

Nói chung, để liệt kê nội dung của một thư mục bao gồm cả các thư mục con của nó, hãy sử dụng:

```
ls -R DIRECTORY_NAME
```

Việc thêm dấu gạch chéo vào `DIRECTORY_NAME` sẽ không có tác dụng:

```
$ ls -R animal
```

cũng tương tự như

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

Sao chép Đệ quy với `cp -r`

`cp -r` (hoặc `-R` hoặc `--recursive`) sẽ cho phép ta sao chép một thư mục cùng với tất cả các thư mục con và tệp của nó.

```
$ 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
```

Trong danh sách trên, chúng ta có thể thấy rằng việc cố gắng sao chép `mydir` vào `newcopy` bằng `cp` mà không có `-r` sẽ khiến hệ thống hiển thị thông báo `cp: omitting directory 'mydir'` (bỏ qua thư mục 'mydir'`). Tuy nhiên, bằng cách thêm tùy chọn `-r`, tất cả nội dung của `mydir` (bao gồm cả chính nó) sẽ được sao chép vào `newcopy`.

Để sao chép các thư mục và thư mục con, hãy sử dụng:

```
cp -r SOURCE DESTINATION
```

Xóa Đệ quy với `rm -r`

`rm -r` sẽ xóa một thư mục và tất cả nội dung của nó (thư mục con và tệp).

WARNING

Hãy hết sức cẩn thận với `-r` hoặc tổ hợp tùy chọn của `-rf` khi sử dụng với lệnh `rm`. Lệnh xóa đệ quy trên một thư mục hệ thống quan trọng có thể sẽ khiến hệ thống không thể sử dụng được. Hãy chỉ sử dụng lệnh xóa đệ quy khi bạn hoàn toàn chắc chắn rằng nội dung của một thư mục là an toàn và có thể bị xóa khỏi máy tính.

Khi cố gắng xóa một thư mục mà không sử dụng `-r`, hệ thống sẽ báo lỗi:

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

Ta phải thêm `-r` như trong lệnh thứ hai để việc xóa có hiệu lực.

NOTE

Có thể bạn sẽ thắc mắc tại sao chúng ta không sử dụng `rmdir` trong trường hợp này. Có một sự khác biệt nhỏ giữa hai lệnh. `rmdir` sẽ chỉ xóa thành công nếu thư mục đã cho trống, trong khi `rm -r` có thể xoá bất kể thư mục này có trống hay không.

Hãy thêm tùy chọn `-i` để xác nhận trước khi tệp bị xóa:

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

Hệ thống sẽ nhắc trước khi xóa thư mục `mydir`.

Khớp mẫu khối trong Tệp và Ký tự Đại diện

Khớp mẫu khối trong Tệp là một tính năng được cung cấp bởi vỏ Unix/Linux để biểu thị nhiều tên tệp bằng cách sử dụng các ký tự đặc biệt được gọi là *ký tự đại diện* (wildcards). Ký tự đại diện về cơ bản là các ký hiệu có thể được sử dụng để thay thế cho một hoặc nhiều ký tự (ví dụ như cho phép hiển thị tất cả các tệp bắt đầu bằng chữ cái như `A` hoặc tất cả các tệp kết thúc bằng chữ cái `.conf`).

Ký tự đại diện rất hữu ích vì chúng có thể được sử dụng với các lệnh như `cp`, `ls` hoặc `rm`.

Sau đây là một số ví dụ về khớp mẫu khối trong tệp:

`rm *`

Xóa tất cả các tệp trong thư mục làm việc hiện tại.

`ls l?st`

Liệt kê tất cả các tệp có tên bắt đầu bằng `l`, theo sau là bất kỳ ký tự đơn nào và kết thúc bằng `st`.

`rmdir [a-z]*`

Xóa tất cả các thư mục có tên bắt đầu bằng một chữ cái.

Các loại Ký tự Đại diện

Có ba ký tự có thể được sử dụng làm ký tự đại diện trong Linux:

`*` (dấu hoa thị)

đại diện cho không, một hoặc nhiều lần xuất hiện của bất kỳ ký tự nào.

`?` (dấu chấm hỏi)

đại diện cho một lần xuất hiện của bất kỳ ký tự nào.

`[ ]` (ký tự trong ngoặc vuông)

đại diện cho bất kỳ sự xuất hiện nào của các ký tự được đặt trong ngoặc vuông. Ta có thể sử dụng nhiều loại ký tự khác nhau dù là số, chữ, ký tự đặc biệt khác. Ví dụ: khối `[0-9]` khớp với tất cả các chữ số.

Dấu hoa thị

Dấu hoa thị (`*`) khớp với không, một hoặc nhiều lần xuất hiện của bất kỳ ký tự nào.

Ví dụ:

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

Thao tác này sẽ tìm tất cả các tệp kết thúc bằng `.png` (chẳng hạn như `photo.png`, `cat.png`, `frank.png`). Lệnh `find` sẽ được thảo luận thêm trong bài học sau.

Tương tự như vậy:

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

sẽ liệt kê tất cả các tệp văn bản bắt đầu bằng các ký tự `lpic-`, theo sau là bất kỳ số lượng ký tự nào và kết thúc bằng `.txt` (chẳng hạn như `lpic-1.txt` và `lpic-2.txt`).

Ký tự đại diện dấu hoa thị có thể được sử dụng để thao tác (sao chép, xóa hoặc di chuyển) với toàn bộ nội dung của một thư mục:

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

Trong ví dụ này, tất cả nội dung của `animal` đã được sao chép vào `forest`.

Nói chung, để sao chép tất cả nội dung của một thư mục, chúng ta sẽ sử dụng:

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

trong đó, `SOURCE_PATH` có thể được bỏ qua nếu chúng ta đã đang ở trong thư mục được yêu cầu.

Dấu hoa thị, cũng giống như bất kỳ ký tự đại diện nào khác, có thể được sử dụng nhiều lần trong cùng một lệnh và tại bất kỳ vị trí nào:

```
$ rm *ate*
```

Tên tệp có tiền tố là không, một hoặc nhiều lần xuất hiện của bất kỳ ký tự nào, theo sau là các chữ cái `ate` và kết thúc bằng không, một hoặc nhiều lần xuất hiện của bất kỳ ký tự nào sẽ bị xóa.

Dấu Hỏi

Dấu chấm hỏi (`?`) khớp với một lần xuất hiện *đơn* của một ký tự.

Hãy xem phần liệt kê sau:

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

Để chỉ trả lại các tệp bắt đầu bằng `l` theo sau bởi bất kỳ ký tự đơn nào và kết thúc `st.txt`, chúng ta sẽ sử dụng ký tự đại diện dấu chấm hỏi (`?`):

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

Chỉ các tệp `last.txt`, `lest.txt` và `list.txt` mới được hiển thị vì chúng khớp với tiêu chí đã cho.

Tương tự như vậy:

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

sẽ cho đầu ra là các tệp có tiền tố là hai ký tự bất kỳ, theo sau là văn bản `st.txt`.

Ký tự trong Ngoặc Vuông

Ký tự đại diện trong ngoặc sẽ tương ứng với bất kỳ sự xuất hiện nào của các ký tự được đặt trong dấu ngoặc vuông.

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

Lệnh này sẽ liệt kê tất cả các tệp bắt đầu bằng `l` và theo sau là *bất kỳ* ký tự nào trong tập hợp `aef` và kết thúc bằng `st.txt`.

Các dấu ngoặc vuông cũng có thể sử dụng với các phạm vi:

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

Lệnh này sẽ xuất ra tất cả các tệp có tên bắt đầu bằng `l` và theo sau là bất kỳ chữ cái viết thường nào trong phạm vi từ `a` đến `z` và kết thúc bằng `st.txt`.

Nhiều phạm vi cũng có thể được áp dụng trong ngoặc vuông:

```
$ ls
student-1A.txt  student-2A.txt  student-3.txt
$ ls student-[0-9][A-Z].txt
student-1A.text student-2A.txt
```

Danh sách hiển thị một thư mục trường học với một danh sách các học sinh đã đăng ký. Để chỉ liệt kê những sinh viên có số báo danh đáp ứng các tiêu chí sau:

- bắt đầu bằng `student-`

- theo sau là một số và một ký tự viết hoa
- và kết thúc bằng `.txt`

Kết hợp các Ký tự Đại diện

Các ký tự đại diện có thể được kết hợp như sau:

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

Thành phần ký tự đại diện đầu tiên (`[plf]`) sẽ khớp với bất kỳ ký tự nào trong số các ký tự `p`, `l` hoặc `f`. Thành phần ký tự đại diện thứ hai (`?`) sẽ khớp với bất kỳ ký tự đơn nào. Thành phần ký tự đại diện thứ ba (`*`) sẽ khớp với không, một hoặc nhiều lần xuất hiện của bất kỳ ký tự nào.

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

Lệnh trên sẽ hiển thị tất cả các tệp bắt đầu bằng chữ cái `f`, theo sau là bất kỳ bộ chữ cái nào, ít nhất một lần xuất hiện của một chữ số và kết thúc bằng `.txt`. Hãy lưu ý rằng `file.txt` sẽ không được hiển thị vì nó không khớp với tiêu chí này.

Bài tập Hướng dẫn

1. Hãy xem danh sách dưới đây:

```
$ 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
```

   - Ký tự `d` biểu thị điều gì trong đầu ra?
   - Tại sao các kích thước lại được đưa ra ở định dạng con người có thể đọc được?
   - Sự khác biệt trong đầu ra sẽ là gì nếu `ls` được sử dụng mà không có đối số?

2. Hãy xem lệnh dưới đây:

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

   - Điều gì sẽ xảy ra với tệp `emp_name` nếu lệnh này được thực thi thành công?
   - Nếu `emp_name` là một thư mục thì tùy chọn nào sẽ được thêm vào `cp` để thực thi lệnh?
   - Nếu `cp` được đổi thành `mv` thì kết quả sẽ như thế nào?

3. Hãy xem danh sách:

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

   Ký tự đại diện nào sẽ giúp ta xóa toàn bộ nội dung của thư mục này?

4. Dựa trên danh sách trên, những tệp nào sẽ được hiển thị bởi lệnh sau?

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

5. Hãy hoàn thành lệnh bằng cách thêm các chữ số và ký tự thích hợp trong dấu ngoặc vuông để liệt kê toàn bộ nội dung ở trên:

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

Bài tập Mở rộng

1. Trong thư mục chính của bạn, hãy tạo các tệp có tên `dog` và `cat`.
2. Vẫn trong thư mục chính của bạn, hãy tạo thư mục có tên `animal`. Hãy di chuyển `dog` và `cat` vào `animal`.
3. Hãy di chuyển đến thư mục `Documents` trong thư mục chính của bạn và tạo thư mục `backup`.
4. Hãy sao chép `animal` và nội dung của nó vào `backup`.
5. Hãy đổi tên `animal` trong `backup` thành `animal.bkup`.
6. Thư mục `/home/lpi/databases` có chứa nhiều tệp bao gồm: `db-1.tar.gz`, `db-2.tar.gz` và `db-3.tar.gz`. Bạn có thể sử dụng lệnh đơn nào để chỉ liệt kê các tệp được đề cập ở trên?

7. Hãy xem danh sách sau:

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

   Với việc sử dụng một ký tự khớp mẫu khối duy nhất, lệnh nào sẽ chỉ xóa các tệp pdf?

Tóm tắt

Trong bài học này, chúng ta đã khám phá cách xem nội dung bên trong một thư mục bằng lệnh `ls`, cách sao chép các tệp và thư mục (`cp`) cũng như cách di chuyển (`mv`) chúng. Chúng ta cũng đã xem xét cách tạo thư mục mới bằng lệnh `mkdir`. Các lệnh để xóa tệp (`rm`) và thư mục (`rmdir`) cũng đã được thảo luận.

Trong bài học này, bạn cũng đã học về khớp mẫu khối tệp và ký tự đại diện. Khớp mẫu khối tệp được sử dụng để biểu thị nhiều tên tệp bằng cách sử dụng các ký tự đặc biệt được gọi là ký tự đại diện. Các ký tự đại diện cơ bản và ý nghĩa của chúng:

`?` (dấu chấm hỏi)
: đại diện cho một lần xuất hiện của một ký tự.

`[ ]` (dấu ngoặc vuông)
: đại diện cho bất kỳ sự xuất hiện nào của các ký tự nằm trong dấu ngoặc vuông.

`*` (dấu hoa thị)
: đại diện cho không, một hoặc nhiều lần xuất hiện của bất kỳ ký tự nào.

Bạn có thể kết hợp bất kỳ ký tự đại diện nào với nhau trong cùng một câu lệnh.

Đáp án Bài tập Hướng dẫn

1. Hãy xem danh sách dưới đây:

```
$ 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
```

   - Ký tự `d` biểu thị điều gì trong đầu ra?

     `d` là ký tự để xác định một thư mục.

   - Tại sao các kích thước lại được đưa ra ở định dạng con người có thể đọc được?

     Do có tùy chọn `-h`.

   - Sự khác biệt trong đầu ra sẽ là gì nếu `ls` được sử dụng mà không có đối số?

     Sẽ chỉ có tên của thư mục và tệp được cung cấp.

2. Hãy xem lệnh dưới đây:

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

   - Điều gì sẽ xảy ra với tệp `emp_name` nếu lệnh này được thực thi thành công?

     `emp_name` sẽ được sao chép vào `backup`.

   - Nếu `emp_name` là một thư mục thì tùy chọn nào sẽ được thêm vào `cp` để thực thi lệnh?

`-r`

- Nếu `cp` được đổi thành `mv` thì kết quả sẽ như thế nào?

  `emp_name` sẽ được di chuyển đến `backup`. Nó sẽ không còn hiện diện trong thư mục chính của người dùng `frank` nữa.

3. Hãy xem danh sách:

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

   Ký tự đại diện nào sẽ giúp ta xóa toàn bộ nội dung của thư mục này?

   Dấu hoa thị `*`.

4. Dựa trên danh sách trên, những tệp nào sẽ được hiển thị bởi lệnh sau?

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

   Tất cả các tệp vì ký tự dấu hoa thị đại diện cho bất kỳ một số lượng ký tự nào.

5. Hãy hoàn thành lệnh bằng cách thêm các chữ số và ký tự thích hợp trong dấu ngoặc vuông để liệt kê toàn bộ nội dung ở trên:

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

   `file[0-9].txt`

Đáp án Bài tập Mở rộng

1. Trong thư mục chính của bạn, hãy tạo các tệp có tên `dog` và `cat`.

```
$ touch dog cat
```

2. Vẫn trong thư mục chính của bạn, hãy tạo thư mục có tên `animal`. Hãy di chuyển `dog` và `cat` vào `animal`.

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

3. Hãy di chuyển đến thư mục `Documents` trong thư mục chính của bạn và tạo thư mục `backup`.

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

4. Hãy sao chép `animal` và nội dung của nó vào `backup`.

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

5. Hãy đổi tên `animal` trong `backup` thành `animal.bkup`.

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

6. Thư mục `/home/lpi/databases` có chứa nhiều tệp bao gồm: `db-1.tar.gz`, `db-2.tar.gz` và `db-3.tar.gz`. Bạn có thể sử dụng lệnh đơn nào để chỉ liệt kê các tệp được đề cập ở trên?

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

7. Hãy xem danh sách sau:

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

   Với việc sử dụng một ký tự khớp mẫu khối duy nhất, lệnh nào sẽ chỉ xóa các tệp pdf?

```
$ rm *.pdf
```

103.3 Bài 2

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.3 Thực hiện Quản lý Tệp cơ bản
Bài:	2 trên 2

Giới thiệu

Cách tìm Tệp

Khi chúng ta sử dụng máy, các tệp sẽ tăng dần về số lượng và kích thước khiến việc định vị một tệp cụ thể đôi khi trở nên khó khăn hơn. May mắn thay, Linux có lệnh `find` cho phép chúng ta nhanh chóng tìm kiếm và định vị các tệp. Lệnh `find` sử dụng cú pháp như sau:

```
find STARTING_PATH OPTIONS EXPRESSION
```

`STARTING_PATH` (Đường dẫn bắt đầu)

xác định thư mục bắt đầu tìm kiếm.

`OPTIONS` (Tuỳ chọn)

kiểm soát hành vi và thêm các tiêu chí cụ thể để tối ưu hóa quá trình tìm kiếm.

`EXPRESSION` (Biểu thức)

xác định truy vấn tìm kiếm.

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

Đường dẫn bắt đầu trong trường hợp này là thư mục hiện tại. Tùy chọn `-name` đã chỉ định rằng phiên tìm kiếm sẽ dựa trên tên của tệp. `myfile.txt` là tên của tệp cần tìm kiếm. Khi sử dụng tính năng khớp mã khối tệp, hãy đảm bảo biểu thức được nằm trong dấu trích dẫn kép:

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

Lệnh này sẽ tìm tất cả các tệp kết thúc bằng `.png` bắt đầu từ thư mục `/home/frank/` trở xuống. Nếu chưa hiểu về cách sử dụng ký tự đại diện dấu hoa thị (`*`), hãy xem lại bài học trước.

Sử dụng tiêu chí để tăng tốc độ tìm kiếm

Chúng ta có thể sử dụng `find` để định vị tệp dựa trên *loại, kích thước* hoặc *thời gian*. Bằng cách chỉ định một hoặc nhiều tùy chọn, chúng ta sẽ thu được kết quả mong muốn trong một khoảng thời gian ngắn hơn.

Các khoá chuyển để tìm tệp dựa trên loại bao gồm:

`-type f`
: tìm kiếm tệp.

`-type d`
: tìm kiếm thư mục.

`-type l`
: tìm kiếm liên kết tượng trưng.

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

Lệnh này sẽ tìm tất cả các thư mục trong thư mục hiện tại và ở dưới có tên `example`.

Các tiêu chí khác có thể được sử dụng với `find` bao gồm:

`-name`
: thực hiện tìm kiếm dựa trên tên đã cho.

`-iname`

tìm kiếm dựa trên tên nhưng không phân biệt chữ hoa và chữ thường (nghĩa là trường hợp của `myFile` cũng sẽ tương tự như `MYFILE`).

`-not`

trả về những kết quả *không* khớp với trường hợp thử nghiệm.

`-maxdepth N`

tìm kiếm trong thư mục hiện tại cũng như các thư mục con ở cấp độ `N` sâu.

Định vị Tệp theo thời gian sửa đổi

`find` cũng cho phép chúng ta lọc phân cấp thư mục dựa trên thời điểm tệp được sửa đổi:

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

Lệnh này sẽ tìm kiếm tất cả các tệp trong toàn bộ hệ thống tệp (đường dẫn bắt đầu sẽ là thư mục gốc, tức `/`) được kết thúc bằng các ký tự `.conf` và đã được sửa đổi trong bảy ngày qua. Lệnh này sẽ yêu cầu các đặc quyền nâng cao để truy cập các thư mục bắt đầu từ cơ sở cấu trúc thư mục của hệ thống (vì thế nên phải sử dụng `sudo` ở đây). Đối số được truyền đến `mtime` biểu thị *số ngày* kể từ khi tệp được sửa đổi lần cuối.

Định vị Tệp theo kích thước

`find` cũng có thể định vị tệp theo *kích thước*. Ví dụ: để tìm kiếm các tệp lớn hơn `2G` trong `/var`:

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

Tùy chọn `-size` sẽ hiển thị các tệp có kích thước tương ứng với đối số được truyền. Một số đối số ví dụ bao gồm:

`-size 100b`

tệp có kích cỡ chính xác là 100 byte.

`-size +100k`

tệp có kích cỡ lớn hơn 100 kilobyte.

`-size -20M`
: tệp có kích cỡ nhỏ hơn 20 megabyte.

`-size +2G`
: tệp có kích cỡ lớn hơn 2 gigabyte.

> **NOTE** Để tìm các tệp trống, chúng ta có thể sử dụng: `find . -size 0b` hoặc `find . -empty`.

Thao tác trên Tập kết quả

Sau khi phiên tìm kiếm hoàn thành, ta có thể thực hiện một hành động trên tập hợp kết quả bằng cách sử dụng `-exec`:

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

Lệnh này sẽ lọc mọi đối tượng bên trong thư mục hiện tại (`.`) và theo xuống dưới để tìm tên tệp kết thúc bằng `.conf`, sau đó thực thi lệnh `chmod 644` để sửa đổi quyền truy cập tệp trên kết quả.

Hiện tại, bạn không cần bận tâm đến ý nghĩa của `'{}' \;` vì nó sẽ được thảo luận sau.

Sử dụng `grep` để lọc Tệp dựa trên nội dung

`grep` được sử dụng để tìm kiếm sự xuất hiện của một từ khóa.

Hãy xem một tình huống mà chúng ta tìm tệp dựa trên nội dung:

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

Lệnh này sẽ tìm kiếm mọi đối tượng trong hệ thống phân cấp thư mục hiện tại (`.`) là một tệp (`-type f`) và sau đó thực thi lệnh `grep "lpi"` cho mọi tệp thỏa mãn các điều kiện. Các tệp phù hợp với các điều kiện này sẽ được in trên màn hình (`-print`). Dấu ngoặc nhọn (`{}`) là phần giữ chỗ cho kết quả trùng khớp với `find`. `{}` được đặt trong dấu trích dẫn đơn (`'`) để tránh truyền các tệp `grep` có tên chứa các ký tự đặc biệt. Lệnh `-exec` sẽ được kết thúc bằng dấu chấm phẩy (`;`). Dấu chấm phẩy này phải được thoát (`\;`) để tránh bị thông dịch bởi vỏ.

Việc thêm tùy chọn `-delete` vào cuối biểu thức sẽ xóa tất cả các tệp khớp. Tùy chọn này chỉ nên

được sử dụng khi người dùng chắc chắn rằng kết quả sẽ chỉ khớp với các tệp mà mình muốn xóa.

Trong ví dụ bên dưới, `find` sẽ định vị tất cả các tệp trong hệ thống phân cấp bắt đầu từ thư mục hiện tại, sau đó xóa tất cả các tệp kết thúc bằng ký tự `.bak`:

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

Lưu trữ Tệp

Lệnh `tar` (Lưu trữ và Nén)

Lệnh `tar` (viết tắt của "tape archive(r)" - lưu trữ dạng băng) được sử dụng để tạo các kho lưu trữ tar bằng cách chuyển đổi một nhóm tệp thành một kho lưu trữ. Các kho lưu trữ được tạo để dễ dàng di chuyển hoặc sao lưu một nhóm tệp. Hãy coi `tar` giống như một công cụ tạo ra chất keo kết dính hoặc nhóm các tệp lại với nhau để có thể di chuyển dễ dàng.

`tar` cũng có khả năng trích xuất kho lưu trữ tar, hiển thị danh sách các tệp có trong kho lưu trữ cũng như thêm các tệp bổ sung vào kho lưu trữ hiện có.

Cú pháp lệnh `tar` sẽ như sau:

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

`OPERATION`

Chỉ cho phép và yêu cầu một đối số hoạt động duy nhất. Các hoạt động được sử dụng thường xuyên nhất là:

`--create` (`-c`)

Tạo một kho lưu trữ tar mới.

`--extract` (`-x`)

Trích xuất toàn bộ kho lưu trữ hoặc một hoặc nhiều tệp từ một kho lưu trữ.

`--list` (`-t`)

Hiển thị danh sách các tệp có trong kho lưu trữ.

`OPTION`

Các tùy chọn được sử dụng thường xuyên nhất là:

`--verbose` (`-v`)

Hiển thị các tệp đang được xử lý bởi lệnh `tar`.

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

Chỉ định tên tệp lưu trữ.

`ARCHIVE_NAME`

Tên của kho lưu trữ.

`FILE_NAME(S)`

Danh sách tên tệp được phân tách bằng dấu cách. Nếu không được cung cấp, toàn bộ kho lưu trữ sẽ được trích xuất.

Tạo Kho Lưu trữ

Giả sử chúng ta có một thư mục có tên là `stuff` nằm trong thư mục hiện tại và muốn lưu nó vào một tệp có tên `archive.tar`. Chúng ta có thể chạy lệnh sau:

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

Sau đây là ý nghĩa thực sự của những khoá chuyển trên:

`-c`

Tạo một kho lưu trữ.

`-v`

Hiển thị tiến trình trong cửa sổ dòng lệnh trong khi tạo kho lưu trữ, còn được gọi là chế độ "verbose". `-v` không phải là tuỳ chọn bắt buộc đối với các lệnh này nhưng nó lại rất hữu ích.

`-f`

Cho phép chỉ định tên tệp của kho lưu trữ.

Nói chung, để lưu trữ một thư mục hoặc một tệp trên Linux, chúng ta sử dụng:

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

NOTE `tar` hoạt động theo cách đệ quy. Nó sẽ thực hiện hành động cần thiết trên mọi thư mục tiếp theo bên trong thư mục được chỉ định.

Để lưu trữ nhiều thư mục cùng một lúc, chúng ta sẽ liệt kê tất cả các thư mục và phân định chúng bằng khoảng trắng trong phần `/PATH/TO/DIRECTORY-OR-FILE`:

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

Điều này sẽ tạo ra kho lưu trữ `stuff1` và `stuff2` trong `archive.tar`

Giải nén một Kho Lưu trữ

Chúng ta có thể giải nén một kho lưu trữ bằng cách sử dụng `tar`:

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

Lệnh này sẽ giải nén nội dung của `archive.tar` vào thư mục hiện tại.

Lệnh này giống như lệnh tạo kho lưu trữ được sử dụng ở trên, ngoại trừ việc khóa chuyển `-x` sẽ thay thế khóa chuyển `-c`.

Để giải nén nội dung của kho lưu trữ vào một thư mục cụ thể, chúng ta sẽ sử dụng `-C`:

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

Lệnh này sẽ giải nén nội dung của `archive.tar` vào thư mục hiện tại.

```
$ ls /tmp
stuff
```

Nén bằng `tar`

Lệnh GNU `tar` đi kèm với các bản phân phối Linux có thể tạo một kho lưu trữ `.tar` và sau đó nén nó bằng cách nén `gzip` hoặc `bzip2` trong một lệnh duy nhất:

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

Lệnh này sẽ tạo một tệp nén bằng thuật toán `gzip` (`-z`).

Mặc dù tính năng nén `gzip` được sử dụng thường xuyên nhất để tạo các tệp `.tar.gz` hoặc `.tgz`, `tar` cũng hỗ trợ cả tính năng nén `bzip2`. Điều này cho phép ta tạo các tệp nén `bzip2` thường được đặt tên là `.tar.bz2`, `.tar.bz` hoặc `.tbz`.

Để làm như vậy, chúng ta sẽ thay thế `-z` cho `gzip` bằng `-j` cho `bzip2`:

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

Để giải nén tệp, chúng ta sẽ thay thế `-c` bằng `-x` (viết tắt của “extract”):

```
$ tar -xzvf archive.tar.gz
```

`gzip` nhanh hơn nhưng nó thường sẽ nén ít hơn và vì vậy nên tệp nén sẽ lớn hơn một chút. `bzip2` chậm hơn nhưng nén nhiều hơn nên tệp nén sẽ nhỏ hơn. Tuy nhiên, `gzip` và `bzip2` nhìn chung trên thực tế là giống nhau và cũng hoạt động tương tự như nhau.

Ngoài ra, chúng ta có thể áp dụng nén `gzip` hoặc `bzip2` bằng cách sử dụng lệnh `gzip` để nén `gzip` và lệnh `bzip` để nén `bzip`. Ví dụ: để áp dụng nén `gzip`, hãy sử dụng:

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

`gzip`
: tạo tệp nén có cùng tên nhưng có đuôi `.gz`.

`gzip`
: xóa các tệp gốc sau khi tạo tệp nén.

Lệnh `bzip2` cũng hoạt động theo cách tương tự.

Để giải nén các tệp, chúng ta có thể sử dụng `gunzip` hoặc `bunzip2` tùy thuộc vào thuật toán được sử dụng để nén tệp.

Lệnh `cpio`

`cpio` là viết tắt của “copy in, copy out” (sao chép vào, sao chép ra). Nó được sử dụng để xử lý các tệp lưu trữ như `*.cpio` hoặc `*.tar`.

`cpio` thực hiện các thao tác sau:

- Sao chép tệp vào một kho lưu trữ.

- Trích xuất tệp từ một kho lưu trữ.

Nó sẽ lấy danh sách các tệp từ đầu vào tiêu chuẩn (hầu hết là đầu ra từ `ls`).

Để tạo kho lưu trữ `cpio`, chúng ta sử dụng:

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

Tùy chọn `-o` sẽ hướng dẫn `cpio` tạo đầu ra. Trong trường hợp này, tệp đầu ra được tạo là `archive.cpio`. Lệnh `ls` sẽ liệt kê nội dung của thư mục hiện tại sắp được lưu trữ.

Để trích xuất kho lưu trữ, chúng ta sử dụng:

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

Tùy chọn `-i` được sử dụng để thực hiện giải nén. Tùy chọn `-d` sẽ tạo thư mục đích. Ký tự `<` đại diện cho đầu vào tiêu chuẩn. Tệp đầu vào được giải nén sẽ là `archive.cpio`.

Lệnh dd

`dd` sẽ sao chép dữ liệu từ vị trí này sang vị trí khác. Cú pháp dòng lệnh của `dd` khác với nhiều chương trình Unix khác. Nó sử dụng cú pháp `option=value` cho các tùy chọn dòng lệnh thay vì các định dạng `-option value` hoặc `--option=value` của tiêu chuẩn GNU:

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

Lệnh này sẽ sao chép nội dung của `oldfile` vào `newfile`, trong đó, `if=` là tệp đầu vào và `of=` là tệp đầu ra.

NOTE

Lệnh `dd` thường sẽ không xuất bất kỳ thứ gì ra màn hình cho đến khi lệnh kết thúc. Bằng cách cung cấp tùy chọn `status=progress`, bảng điều khiển sẽ hiển thị khối lượng công việc được thực hiện bởi lệnh. Ví dụ: `dd status=progress if=oldfile of=newfile`.

`dd` cũng được sử dụng để thay đổi dữ liệu thành chữ hoa/chữ thường hoặc ghi trực tiếp vào các thiết bị khối như `/dev/sdb`:

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

Lệnh này sẽ sao chép tất cả nội dung của `oldfile` vào `newfile` và viết hoa toàn bộ văn bản.

Lệnh sau sẽ sao lưu toàn bộ ổ cứng nằm tại `/dev/sda` vào một tệp có tên `backup.dd`:

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

Bài tập Hướng dẫn

1. Hãy xem danh sách sau:

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

   - Lệnh này sẽ xuất ra loại tệp nào?

   - Việc tìm kiếm sẽ bắt đầu từ thư mục nào?

2. Một người dùng muốn nén thư mục sao lưu của mình. Anh ta sử dụng lệnh sau:

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

   Tùy chọn nào đang bị thiếu để có thể nén bản sao lưu bằng thuật toán `gzip`?

Bài tập Mở rộng

1. Là quản trị viên hệ thống, bạn cần phải thực hiện việc kiểm tra thường xuyên để xóa các tệp có dung lượng lớn. Các tệp đồ sộ này nằm trong `/var` và kết thúc bằng phần mở rộng `.backup`.
   - Hãy viết lệnh sử dụng `find` để định vị các tệp này:
   - Một phân tích về kích thước của các tệp này cho thấy chúng có phạm vi từ `100M` đến `1000M`. Hãy hoàn thành lệnh trên với thông tin mới này để có thể định vị các tệp sao lưu từ `100M` đến `1000M`:
   - Cuối cùng, hãy hoàn thành lệnh này với hành động xóa để các tệp này sẽ bị xóa:
2. Trong thư mục `/var` có tồn tại bốn tệp sao lưu sau:

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

   - Bằng cách sử dụng `tar`, hãy chỉ định lệnh sẽ tạo tệp lưu trữ có tên `db-first-quý-2018.backup.tar`:
   - Bằng cách sử dụng `tar`, hãy chỉ định lệnh sẽ tạo kho lưu trữ và nén nó bằng `gzip`. Hãy lưu ý rằng tên tệp kết quả phải kết thúc bằng `.gz`:

Tóm tắt

Trong bài học này, chúng ta đã học về:

- Cách tìm tệp bằng `find`.
- Cách thêm tiêu chí tìm kiếm dựa trên thời gian, loại tệp hoặc kích thước bằng cách cung cấp đối số cho `find`.
- Làm thế nào để thao tác trên một tập hợp kết quả.
- Cách lưu trữ, nén và giải nén tệp bằng `tar`.
- Xử lý kho lưu trữ với `cpio`.
- Sao chép tệp với `dd`.

Đáp án Bài tập Hướng dẫn

1. Hãy xem phần liệt kê sau:

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

   - Lệnh này sẽ xuất ra loại tệp nào?

     Các thư mục.

   - Việc tìm kiếm sẽ bắt đầu từ thư mục nào?

     `/home/frank/Documents`

2. Một người dùng muốn nén thư mục sao lưu của mình. Anh ta sử dụng lệnh sau:

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

   Tùy chọn nào đang bị thiếu để có thể nén bản sao lưu bằng thuật toán `gzip`?

   Tuỳ chọn `-z`.

Đáp án Bài tập Mở rộng

1. Là quản trị viên hệ thống, bạn cần phải thực hiện việc kiểm tra thường xuyên để xóa các tệp có dung lượng lớn. Các tệp đồ sộ này nằm trong `/var` và kết thúc bằng phần mở rộng `.backup`.
   - Hãy viết lệnh sử dụng `find` để định vị các tệp này:

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

   - Một phân tích về kích thước của các tệp này cho thấy chúng có phạm vi từ `100M` đến `1000M`. Hãy hoàn thành lệnh trên với thông tin mới này để có thể định vị các tệp sao lưu từ `100M` đến `1000M`:

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

   - Cuối cùng, hãy hoàn thành lệnh này với hành động xóa để các tệp này sẽ bị xóa:

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

2. Trong thư mục `/var` có tồn tại bốn tệp sao lưu sau:

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

   - Bằng cách sử dụng `tar`, hãy chỉ định lệnh sẽ tạo tệp lưu trữ có tên `db-first-quý-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
     ```

   - Bằng cách sử dụng `tar`, hãy chỉ định lệnh sẽ tạo kho lưu trữ và nén nó bằng `gzip`. Hãy lưu ý rằng tên tệp kết quả phải kết thúc bằng `.gz`:

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

103.4 Sử dụng Luồng, Đường ống và Chuyển hướng

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 103.4

Khối lượng

4

Các lĩnh vực kiến thức chính

- Chuyển hướng đầu vào tiêu chuẩn, đầu ra tiêu chuẩn và lỗi tiêu chuẩn.
- Chuyển đầu ra của một lệnh sang đầu vào của lệnh khác.
- Sử dụng đầu ra của một lệnh làm đối số cho lệnh khác.
- Gửi đầu ra tới cả đầu ra tiêu chuẩn và tệp.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

- `tee`
- `xargs`

Linux Professional Institute

103.4 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.4 Sử dụng Luồng, Đường ống và Chuyển hướng
Bài:	1 trên 2

Giới thiệu

Tất cả các chương trình máy tính đều tuân theo cùng một nguyên tắc chung: dữ liệu nhận được từ một số nguồn sẽ được chuyển đổi để tạo ra một kết quả dễ hiểu. Trong ngữ cảnh của vỏ Linux, nguồn dữ liệu có thể là tệp cục bộ, tệp từ xa, thiết bị (như bàn phím), v.v. Đầu ra của chương trình thường được hiển thị trên màn hình, nhưng việc lưu trữ dữ liệu đầu ra trong một hệ thống tệp cục bộ, gửi nó đến một thiết bị từ xa, phát nó qua loa âm thanh, v.v. cũng là khá phổ biến.

Các hệ điều hành lấy cảm hứng từ Unix (như Linux) cung cấp rất nhiều các phương thức xuất/nhập. Đặc biệt, phương thức *mô tả tệp* (file descriptors) cho phép chúng ta liên kết động các số nguyên với các kênh dữ liệu để một tiến trình có thể tham chiếu chúng dưới dạng các luồng dữ liệu đầu vào/đầu ra.

Các tiến trình Linux tiêu chuẩn có ba kênh giao tiếp được mở theo mặc định: kênh đầu vào *tiêu chuẩn* (hầu hết được gọi đơn giản là *stdin*), kênh đầu ra *tiêu chuẩn* (*stdout*) và kênh *lỗi tiêu chuẩn* (*stderr*). Các mô tả tệp bằng số được gán cho các kênh này là `0` cho stdin, `1` cho stdout và `2` cho stderr. Các kênh giao tiếp cũng có thể được truy cập thông qua các thiết bị đặc biệt là `/dev/stdin`, `/dev/stdout` và `/dev/stderr`.

Ba kênh giao tiếp tiêu chuẩn này cho phép các lập trình viên viết mã đọc và ghi dữ liệu mà không phải lo lắng xem nó đến từ hoặc sẽ đi tới loại phương tiện nào. Ví dụ: nếu một chương trình cần một tập hợp dữ liệu làm đầu vào, chương trình đó chỉ cần yêu cầu dữ liệu từ đầu vào tiêu chuẩn và khi đó bất kỳ thứ gì đang được sử dụng làm đầu vào tiêu chuẩn sẽ cung cấp dữ liệu đó. Tương tự như vậy, phương pháp đơn giản nhất mà một chương trình có thể sử dụng để hiển thị đầu ra là ghi nó vào đầu ra tiêu chuẩn. Trong một phiên vỏ tiêu chuẩn, bàn phím được xác định là stdin và màn hình điều khiển được xác định stdout và stderr.

Vỏ Bash có khả năng gán lại các kênh liên lạc khi tải chương trình (ví dụ: nó cho phép ghi đè màn hình làm đầu ra tiêu chuẩn và sử dụng một tệp trong hệ thống tệp cục bộ làm stdout).

Chuyển hướng

Việc gán lại bộ mô tả tệp của một kênh trong môi trường vỏ được gọi là một phiên *chuyển hướng*. Phiên chuyển hướng được xác định bởi một ký tự đặc biệt trong dòng lệnh. Ví dụ: để chuyển hướng đầu ra tiêu chuẩn của một tiến trình sang một tệp, ký hiệu *lớn hơn* `>` sẽ được đặt ở cuối lệnh và theo sau là đường dẫn đến tệp sẽ nhận đầu ra được chuyển hướng:

```
$ cat /proc/cpuinfo >/tmp/cpu.txt
```

Theo mặc định, chỉ có nội dung đến stdout mới được chuyển hướng. Điều này xảy ra vì giá trị bằng số của bộ mô tả tệp phải được chỉ định ngay trước ký hiệu lớn hơn và khi không được chỉ định, Bash sẽ chuyển hướng đầu ra tiêu chuẩn. Do đó, việc sử dụng `>` sẽ tương đương với việc sử dụng `1>` (giá trị bộ mô tả tệp của stdout là `1`).

Để nắm bắt nội dung của stderr, ta nên sử dụng chuyển hướng `2>`. Hầu hết các chương trình dòng lệnh đều gửi thông tin gỡ lỗi và thông báo lỗi đến kênh lỗi tiêu chuẩn. Ví dụ: để ghi lại thông báo lỗi được kích hoạt khi ta cố đọc một tệp không tồn tại:

```
$ cat /proc/cpu_info 2>/tmp/error.txt
$ cat /tmp/error.txt
cat: /proc/cpu_info: No such file or directory
```

Cả stdout và stderr đều được chuyển hướng đến cùng một mục tiêu bằng `&>` hoặc `>&`. Điều quan trọng là không đặt bất kỳ khoảng trắng nào bên cạnh dấu `&`; nếu có, Bash sẽ coi đó là hướng dẫn để chạy tiến trình ở chế độ nền và không thực hiện chuyển hướng.

Mục tiêu phải là đường dẫn đến tệp có thể ghi được, chẳng hạn như `/tmp/cpu.txt` hoặc bộ mô tả tệp có thể ghi. Mục tiêu của bộ mô tả tệp sẽ được biểu thị bằng dấu `&`, theo sau là giá trị bằng số của bộ mô tả tệp. Ví dụ: `1>&2` sẽ chuyển hướng từ stdout sang stderr. Để làm ngược lại, tức là từ

stderr sang stdout, ta sẽ sử dụng `2>&1`.

Mặc dù không hữu ích lắm vì có một cách ngắn hơn để thực hiện cùng một tác vụ, chúng ta vẫn có thể chuyển hướng từ stderr sang stdout và sau đó chuyển hướng nó đến một tệp. Ví dụ: một phiên chuyển hướng để ghi cả stderr và stdout vào một tệp có tên `log.txt` có thể được viết là `>log.txt 2>&1`. Tuy nhiên, lý do chính để chuyển hướng từ stderr sang stdout là để cho phép phân tích cú pháp gỡ lỗi và thông báo lỗi. Ta có thể chuyển hướng đầu ra tiêu chuẩn của một chương trình sang đầu vào tiêu chuẩn của chương trình khác, nhưng ta không thể chuyển hướng trực tiếp lỗi tiêu chuẩn sang đầu vào tiêu chuẩn của một chương trình khác. Do đó, các thông báo của chương trình được gửi đến stderr trước tiên cần được chuyển hướng đến stdout để stdin của chương trình khác có thể đọc được.

Để loại bỏ đầu ra của một lệnh, nội dung của nó có thể được chuyển hướng đến tệp đặc biệt `/dev/null`. Ví dụ: `>log.txt 2>/dev/null` lưu nội dung của stdout trong tệp `log.txt` và loại bỏ stderr. Tệp `/dev/null` có thể được ghi bởi bất kỳ người dùng nào, nhưng ta sẽ không thể khôi phục dữ liệu từ tệp này vì tệp không được lưu trữ ở bất kỳ một vị trí nào.

Một thông báo lỗi sẽ được hiển thị nếu mục tiêu đã chỉ định không thể ghi được (nếu đường dẫn trỏ đến một thư mục hoặc tệp chỉ cho phép đọc) và không có sửa đổi nào đối với mục tiêu được thực hiện. Tuy nhiên, một phiên chuyển hướng đầu ra sẽ ghi đè lên một mục tiêu có thể ghi hiện có mà không có bất kỳ xác nhận nào. Các tệp sẽ bị ghi đè bởi chuyển hướng đầu ra trừ khi tùy chọn Bash `noclobber` được kích hoạt. Việc này có thể thực hiện trong phiên hiện tại bằng lệnh `set -o noclobber` hoặc `set -C`:

```
$ set -o noclobber
$ cat /proc/cpu_info 2>/tmp/error.txt
-bash: /tmp/error.txt: cannot overwrite existing file
```

Để huỷ tùy chọn `noclobber` cho phiên hiện tại, hãy chạy `set +o noclobber` hoặc `set +C`. Để duy trì tùy chọn `noclobber`, tùy chọn này phải được đưa vào hồ sơ Bash của người dùng hoặc trong hồ sơ toàn hệ thống.

Ngay cả khi bật tùy chọn `noclobber`, ta vẫn có thể nối thêm dữ liệu được chuyển hướng vào nội dung hiện có. Điều này được thực hiện với một chuyển hướng được viết bằng hai ký hiệu lớn hơn `>>`:

```
$ 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
```

Ngay cả khi bật tùy chọn `noclobber`, ta vẫn có thể nối thêm dữ liệu được chuyển hướng vào nội dung hiện có. Điều này được thực hiện với một chuyển hướng được viết bằng hai ký hiệu lớn hơn (`>>`):

Nguồn dữ liệu đầu vào tiêu chuẩn của một tiến trình cũng có thể được chỉ định lại. Ký hiệu nhỏ hơn (`<`) được sử dụng để chuyển hướng nội dung của tệp sang stdin của một tiến trình. Trong trường hợp này, dữ liệu sẽ đi từ phải sang trái: bộ mô tả được gán lại được coi là 0 ở bên trái của ký hiệu nhỏ hơn và nguồn dữ liệu (đường dẫn đến tệp) phải ở bên phải của ký hiệu nhỏ hơn. Lệnh `uniq`, giống như hầu hết các tiện ích dòng lệnh dùng để xử lý văn bản, chấp nhận dữ liệu được gửi tới stdin theo mặc định:

```
$ uniq -c </tmp/error.txt
      2 cat: /proc/cpu_info: No such file or directory
```

Tùy chọn `-c` sẽ khiến `uniq` hiển thị số lần một dòng lặp lại xuất hiện trong văn bản. Vì giá trị bằng số của bộ mô tả tệp được chuyển hướng đã được nén, lệnh mẫu sẽ tương đương với `uniq -c 0</tmp/error.txt`. Việc sử dụng một bộ mô tả tệp khác `0` trong một phiên chuyển hướng đầu vào sẽ chỉ có ý nghĩa trong một số ngữ cảnh cụ thể vì chương trình có thể yêu cầu dữ liệu tại các bộ mô tả tệp `3`, `4`, v.v. Các chương trình có thể sử dụng bất kỳ số nguyên nào lớn hơn 2 để làm bộ mô tả tệp mới cho đầu vào/đầu ra dữ liệu. Ví dụ: mã C sau đây sẽ đọc dữ liệu từ bộ mô tả tệp `3` và chỉ sao chép nó vào bộ mô tả tệp `4`:

NOTE Chương trình phải xử lý các bộ mô tả tệp như vậy một cách chính xác. Nếu không, chương trình có thể sẽ thực hiện một thao tác đọc hoặc ghi không hợp lệ và gặp phải sự cố.

```
#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);
}
```

Để kiểm tra, hãy lưu mã mẫu dưới dạng `fd.c` và biên dịch mã đó với `gcc -o fd fd.c`. Chương trình này cần có các bộ mô tả tệp 3 và 4 để có thể được đọc và ghi vào chúng. Ví dụ: tệp `/tmp/error.txt` được tạo trước đó có thể được sử dụng làm nguồn cho bộ mô tả tệp `3` và bộ mô tả tệp `4` có thể được chuyển hướng đến 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
```

Từ góc nhìn của lập trình viên, việc sử dụng bộ mô tả tệp sẽ tránh được việc xử lý các đường dẫn hệ thống tệp và phân tích cú pháp tùy chọn. Một bộ mô tả tệp thậm chí có thể được sử dụng làm cả đầu vào và đầu ra. Trong trường hợp này, bộ mô tả tệp được xác định trong dòng lệnh sẽ có cả ký hiệu nhỏ hơn và lớn hơn (như trong `3<>/tmp/error.txt`).

Here Document và Here String

Một cách khác để chuyển hướng đầu vào sẽ liên quan đến các phương thức *Here document* và *Here string*. Phiên chuyển hướng Here document sẽ cho phép nhập văn bản nhiều dòng được sử dụng làm nội dung được chuyển hướng. Hai ký hiệu nhỏ hơn (`<<`) biểu thị cho một phiên chuyển hướng Here document:

```
$ wc -c <<EOF
> How many characters
> in this Here document?
> EOF
43
```

Ở bên phải của hai ký hiệu nhỏ hơn `<<` là thuật ngữ kết thúc `EOF`. Chế độ chèn sẽ kết thúc ngay sau khi một dòng chỉ chứa duy nhất cụm từ kết thúc được nhập. Bất kỳ một thuật ngữ nào khác cũng đều có thể được sử dụng làm thuật ngữ kết thúc, nhưng chúng ta không được đặt các ký tự trống giữa ký hiệu nhỏ hơn và thuật ngữ kết thúc. Trong ví dụ trên, hai dòng văn bản đã được gửi đến stdin của lệnh `wc -c` hiển thị số lượng ký tự. Như với các phiên chuyển hướng đầu vào cho các tệp, stdin (bộ mô tả tệp `0`) sẽ được tự giả định nếu bộ mô tả tệp được chuyển hướng được nén.

Phương thức Here string cũng giống như phương thức Here document nhưng chỉ dành cho một dòng:

```
$ wc -c <<<"How many characters in this Here string?"
41
```

Trong ví dụ này, chuỗi ở bên phải của ba dấu nhỏ hơn được gửi đến stdin của lệnh `wc -c` dùng để đếm số lượng ký tự. Các chuỗi chứa khoảng trắng phải nằm trong dấu trích dẫn kép; nếu không, chỉ có từ đầu tiên được sử dụng làm Here string và các từ còn lại sẽ được chuyển làm đối số cho lệnh.

Bài tập Hướng dẫn

1. Ngoài các tệp văn bản, lệnh `cat` cũng có thể hoạt động với dữ liệu nhị phân, chẳng hạn như gửi nội dung của một thiết bị khối đến một tệp. Bằng cách sử dụng tác vụ chuyển hướng, làm cách nào để `cat` có thể gửi nội dung của thiết bị `/dev/sdc` tới tệp `sdc.img` trong thư mục hiện tại?

2. Tên của kênh tiêu chuẩn được chuyển hướng bằng lệnh `date 1> now.txt` là gì?

3. Sau khi cố gắng ghi đè lên một tệp bằng cách chuyển hướng, người dùng gặp lỗi thông báo rằng tùy chọn `noclobber` đã được kích hoạt. Làm cách nào để vô hiệu hoá `noclobber` cho phiên hiện tại?

4. Kết quả của lệnh `cat <<.>/dev/stdout` sẽ là gì?

Bài tập Mở rộng

1. Lệnh `cat /proc/cpu_info` hiển thị thông báo lỗi vì `/proc/cpu_info` không tồn tại. Lệnh `cat /proc/cpu_info 2>1` sẽ chuyển hướng thông báo lỗi đến đâu?

2. Có thể loại bỏ nội dung được gửi tới `/dev/null` nếu tùy chọn `noclobber` đã được kích hoạt cho phiên vỏ hiện tại không?

3. Nếu không sử dụng `echo`, làm cách nào để chuyển hướng nội dung của biến `$USER` đến stdin của lệnh `sha1sum`?

4. Nhân Linux giữ các liên kết tượng trưng trong `/proc/PID/fd/` cho mọi tệp được mở bởi một tiến trình, trong đó, *PID* là số nhận dạng của tiến trình tương ứng. Làm cách nào để quản trị viên hệ thống có thể sử dụng thư mục đó để xác minh vị trí của các tệp nhật ký được mở bởi `nginx` (giả sử PID của nó là `1234`)?

5. Có thể thực hiện các phép tính số học chỉ bằng cách sử dụng các lệnh tích hợp sẵn của vỏ, nhưng các phép tính số chấm động (floating point) yêu cầu các chương trình cụ thể như `bc` (basic calculator - *máy tính cơ bản*). Với `bc`, ta thậm chí có thể chỉ định số vị trí thập phân với tham số `scale`. Tuy nhiên, `bc` chỉ chấp nhận các hoạt động thông qua đầu vào tiêu chuẩn của nó thường được nhập trong chế độ tương tác. Bằng cách sử dụng chuỗi Here, làm cách nào để có thể gửi thao tác số chấm động `scale=6; 1/3` đến đầu vào tiêu chuẩn của `bc`?

Tóm tắt

Bài học này bao gồm các phương pháp để chạy một chương trình và chuyển hướng các kênh giao tiếp tiêu chuẩn của nó. Các tiến trình Linux sử dụng các kênh tiêu chuẩn này làm _ bộ mô tả tệp_ chung để đọc và ghi dữ liệu, giúp ta tùy ý thay đổi chúng thành tệp hoặc thiết bị. Bài học đã đi theo các bước sau:

- Bộ mô tả tệp là gì và vai trò của chúng trong Linux.
- Các kênh giao tiếp tiêu chuẩn của mọi tiến trình: *stdin*, *stdout* và *stderr*.
- Cách thực hiện chính xác một lệnh bằng cách sử dụng chuyển hướng dữ liệu cho cả đầu vào và đầu ra.
- Cách sử dụng *Here Documents* và *Here Strings* trong chuyển hướng đầu vào.

Các lệnh và tiến trình được nhắc đến là:

- Các toán tử chuyển hướng: `>`, `<`, `>>`, `<<`, `<<<`.
- Lệnh `cat`, `set`, `uniq` và `wc`.

Đáp án Bài tập Hướng dẫn

1. Ngoài các tệp văn bản, lệnh `cat` cũng có thể hoạt động với dữ liệu nhị phân, chẳng hạn như gửi nội dung của một thiết bị khối đến một tệp. Bằng cách sử dụng tác vụ chuyển hướng, làm cách nào để `cat` có thể gửi nội dung của thiết bị `/dev/sdc` tới tệp `sdc.img` trong thư mục hiện tại?

   ```
   $ cat /dev/sdc > sdc.img
   ```

2. Tên của kênh tiêu chuẩn được chuyển hướng bằng lệnh `date 1> now.txt` là gì?

   Đầu ra tiêu chuẩn hoặc stdout.

3. Sau khi cố gắng ghi đè lên một tệp bằng cách chuyển hướng, người dùng gặp lỗi thông báo rằng tùy chọn `noclobber` đã được kích hoạt. Làm cách nào để vô hiệu hoá `noclobber` cho phiên hiện tại?

   `set +C` hoặc `set +o noclobber`

4. Kết quả của lệnh `cat <<.>/dev/stdout` sẽ là gì?

   Bash sẽ vào chế độ nhập Heredoc, sau đó thoát ra khi một dấu chấm xuất hiện trong một dòng. Văn bản đã nhập sẽ được chuyển hướng đến stdout (được in trên màn hình).

Đáp án Bài tập Mở rộng

1. Lệnh `cat /proc/cpu_info` hiển thị thông báo lỗi vì `/proc/cpu_info` không tồn tại. Lệnh `cat /proc/cpu_info 2>1` sẽ chuyển hướng thông báo lỗi đến đâu?

   Đến một tệp có tên `1` trong thư mục hiện tại.

2. Có thể loại bỏ nội dung được gửi tới `/dev/null` nếu tùy chọn `noclobber` đã được kích hoạt cho phiên vỏ hiện tại không?

   Có thể. `/dev/null` là một tệp đặc biệt không bị ảnh hưởng bởi `noclobber`.

3. Nếu không sử dụng `echo`, làm cách nào để chuyển hướng nội dung của biến `$USER` đến stdin của lệnh `sha1sum`?

   ```
   $ sha1sum <<<$USER
   ```

4. Nhân Linux giữ các liên kết tượng trưng trong `/proc/PID/fd/` cho mọi tệp được mở bởi một tiến trình, trong đó, *PID* là số nhận dạng của tiến trình tương ứng. Làm cách nào để quản trị viên hệ thống có thể sử dụng thư mục đó để xác minh vị trí của các tệp nhật ký được mở bởi `nginx` (giả sử PID của nó là `1234`)?

   Bằng cách thực hiện lệnh `ls -l /proc/1234/fd`. Lệnh này sẽ hiển thị đích của mọi liên kết tượng trưng trong thư mục.

5. Có thể thực hiện các phép tính số học chỉ bằng cách sử dụng các lệnh tích hợp sẵn của vỏ, nhưng các phép tính số chấm động (floating point) yêu cầu các chương trình cụ thể như `bc` (basic calculator - *máy tính cơ bản*). Với `bc`, ta thậm chí có thể chỉ định số vị trí thập phân với tham số `scale`. Tuy nhiên, `bc` chỉ chấp nhận các hoạt động thông qua đầu vào tiêu chuẩn của nó thường được nhập trong chế độ tương tác. Bằng cách sử dụng chuỗi Here, làm cách nào để có thể gửi thao tác số chấm động `scale=6; 1/3` đến đầu vào tiêu chuẩn của `bc`?

   ```
   $ bc <<<"scale=6; 1/3"
   ```

103.4 Bài 2

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.4 Sử dụng Luồng, Đường ống và Chuyển hướng
Bài:	2 trên 2

Giới thiệu

Một khía cạnh trong triết lý của Unix đã khẳng định rằng: mỗi một chương trình nên có một mục đích cụ thể và không nên được tích hợp với các tính năng nằm ngoài phạm vi của nó. Tuy nhiên, việc giữ cho mọi thứ đơn giản không có nghĩa là kết quả sẽ ít phức tạp hơn vì nhiều chương trình có thể được kết nối với nhau để tạo ra một kết quả đầu ra tổng hợp. Ký tự thanh dọc `|` (còn được gọi là ký hiệu *ống*) có thể được sử dụng để tạo một đường dẫn nối trực tiếp đầu ra của một chương trình với đầu vào của một chương trình khác, trong khi *trình thay thế lệnh* sẽ cho phép người dùng lưu trữ đầu ra của một chương trình trong một biến hoặc trực tiếp sử dụng nó làm đối số cho một lệnh khác.

Đường ống

Không giống như chuyển hướng, với đường ống, dữ liệu sẽ đi theo luồng từ trái sang phải trong dòng lệnh và mục tiêu là một tiến trình khác chứ không phải một đường dẫn hệ thống tệp, bộ mô tả tệp hay Here document. Ký tự ống `|` sẽ yêu cầu vỏ bắt đầu tất cả các lệnh riêng biệt cùng một lúc và kết nối đầu ra của lệnh trước với đầu vào của lệnh sau, từ trái sang phải. Ví dụ: thay vì sử dụng chuyển hướng, nội dung của tệp `/proc/cpuinfo` được gửi tới đầu ra tiêu chuẩn bởi `cat` có

thể được dẫn tới stdin của `wc` bằng lệnh sau:

```
$ cat /proc/cpuinfo | wc
   208    1184    6096
```

Trong trường hợp không có đường dẫn đến tệp, `wc` sẽ đếm số dòng, từ và ký tự mà nó nhận được từ stdin của nó (như trường hợp trong ví dụ). Có thể có nhiều đường ống trong một lệnh ghép. Trong ví dụ sau, có hai đường ống được sử dụng:

```
$ cat /proc/cpuinfo | grep 'model name' | uniq
model name      : Intel(R) Xeon(R) CPU           X5355  @ 2.66GHz
```

Nội dung của tệp `/proc/cpuinfo` do `cat /proc/cpuinfo` tạo ra đã được dẫn đến lệnh `grep 'model name'`. Lệnh này sau đó sẽ chỉ chọn các dòng có chứa cụm từ `model name`. Vì máy chạy ví dụ có nhiều CPU nên có dòng `model name` đã lặp đi lặp lại. Đường ống cuối cùng sẽ kết nối `grep 'model name'` với `uniq` và chịu trách nhiệm bỏ qua bất kỳ dòng nào tương đương với dòng đã xuất hiện trong kết quả của lệnh `grep`.

Các đường ống có thể được kết hợp với các phiên chuyển hướng trong cùng một dòng lệnh. Ví dụ trước có thể được viết lại thành một dạng đơn giản hơn:

```
$ grep 'model name' </proc/cpuinfo | uniq
model name      : Intel(R) Xeon(R) CPU           X5355  @ 2.66GHz
```

Chuyển hướng đầu vào cho `grep` không thực sự cần thiết vì `grep` chấp nhận đường dẫn tệp làm đối số, nhưng ví dụ đã minh họa cách xây dựng các lệnh kết hợp như vậy.

Các đường ống và phiên chuyển hướng là độc quyền, nghĩa là một nguồn chỉ có thể được ánh xạ tới một mục tiêu duy nhất. Tuy nhiên, ta có thể chuyển hướng đầu ra tới một tệp và vẫn nhìn thấy nó trên màn hình bằng chương trình `tee`. Để làm điều này, chương trình đầu tiên sẽ gửi đầu ra của nó tới stdin của `tee` và tên tệp sẽ được cung cấp cho chương trình sau để lưu trữ dữ liệu:

```
$ 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
```

Đầu ra của chương trình cuối cùng trong chuỗi được tạo bởi `uniq` sẽ được hiển thị và lưu trữ trong tệp `cpu_model.txt`. Để không ghi đè nội dung của tệp được cung cấp và nối thêm dữ liệu

vào tệp, tùy chọn `-a` phải đi cùng với `tee`.

Chỉ có đầu ra tiêu chuẩn của một tiến trình mới được đường ống tiếp nhận. Giả sử người dùng phải trải qua một quá trình biên dịch dài trên màn hình, đồng thời phải lưu cả đầu ra tiêu chuẩn và lỗi tiêu chuẩn vào một tệp để kiểm tra sau. Nếu thư mục hiện tại không có *Makefile*, lệnh sau sẽ xuất ra lỗi:

```
$ make | tee log.txt
make: *** No targets specified and no makefile found.  Stop.
```

Mặc dù được hiển thị trên màn hình nhưng thông báo lỗi do `make` tạo ra không được `tee` ghi lại và tệp *log.txt* đã được tạo trống. Việc chuyển hướng cần phải được thực hiện trước khi một đường ống có thể bắt được 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.
```

Trong ví dụ này, stderr của `make` đã được chuyển hướng đến stdout. Vì vậy, `tee` có thể bắt nó bằng một đường ống, hiển thị nó trên màn hình và lưu nó trong tệp `log.txt`. Trong những trường hợp như vậy, việc lưu các thông báo lỗi để kiểm tra sau sẽ trở nên hữu ích.

Trình thay thế Lệnh

Một phương pháp khác để bắt đầu ra của một lệnh là *trình thay thế lệnh*. Bằng cách đặt một lệnh bên trong dấu trích dẫn ngược (`), Bash sẽ thay thế lệnh bằng đầu ra tiêu chuẩn của nó. Ví dụ sau đây sẽ cho thấy cách sử dụng stdout của một chương trình làm đối số cho một chương trình khác:

```
$ mkdir `date +%Y-%m-%d`
$ ls
2019-09-05
```

Đầu ra của chương trình `date` (ngày hiện tại, được định dạng là *năm-tháng-ngày*) được sử dụng làm đối số để tạo một thư mục với `mkdir`. Chúng ta có thể thu được một kết quả y hệt bằng cách sử dụng `$()` thay vì dấu trích dẫn ngược:

```
$ rmdir 2019-09-05
$ mkdir $(date +%Y-%m-%d)
```

```
$ ls
2019-09-05
```

Phương pháp tương tự có thể được sử dụng để lưu trữ đầu ra của lệnh dưới dạng một biến:

```
$ OS=`uname -o`
$ echo $OS
GNU/Linux
```

Lệnh `uname -o` đã cho ra tên chung của hệ điều hành hiện tại được lưu trữ trong biến phiên `OS`. Việc gán đầu ra của một lệnh cho một biến rất hữu ích trong các tệp lệnh vì nó giúp ta lưu trữ và đánh giá dữ liệu theo nhiều cách khác nhau.

Tùy thuộc vào đầu ra được tạo bởi lệnh được thay thế, trình thay thế lệnh tích hợp sẵn có thể sẽ không phù hợp. Một phương pháp phức tạp hơn để sử dụng đầu ra của một chương trình làm đối số cho một chương trình khác là áp dụng một phương thức trung gian được gọi là `xargs`. Chương trình `xargs` sẽ sử dụng nội dung mà nó nhận được qua stdin để chạy một lệnh nhất định sử dụng nội dung đó làm đối số. Ví dụ sau đây sẽ cho thấy `xargs` đang chạy chương trình `identify` với các đối số do chương trình `find` cung cấp:

```
$ 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
```

Chương trình `identify` là một phần của *ImageMagick* - một bộ công cụ dòng lệnh dùng để kiểm tra, chuyển đổi và chỉnh sửa hầu hết các loại tệp hình ảnh. Trong ví dụ này, `xargs` đã lấy tất cả các đường dẫn được liệt kê bởi `find` và đặt chúng làm đối số cho `identify`, sau đó hiển thị thông tin cho từng tệp được định dạng theo yêu cầu của tùy chọn `-format`. Các tệp được tìm thấy bởi `find` trong ví dụ này là các hình ảnh có chứa biểu tượng phân phối trong hệ thống tệp Debian. `-format` là tham số cho `identify` chứ không phải cho `xargs`.

Tùy chọn `-n 1` yêu cầu `xargs` chạy lệnh đã cho, mỗi lần với một đối số duy nhất. Trong trường hợp của ví dụ, thay vì chuyển tất cả các đường dẫn do `find` tìm thấy dưới dạng danh sách đối số cho `identify`, việc sử dụng `xargs -n 1` sẽ thực thi lệnh `identify` cho từng đường dẫn riêng

biệt. Việc sử dụng `-n 2` sẽ thực thi `identify` với hai đường dẫn làm đối số, `-n 3` với ba đường dẫn làm đối số, v.v. Tương tự, khi `xargs` xử lý nội dung nhiều dòng — như trường hợp với đầu vào do `find` cung cấp — tùy chọn `-L` có thể được sử dụng để giới hạn số lượng dòng sẽ được sử dụng làm đối số cho mỗi lần thực thi lệnh.

NOTE

Việc sử dụng `xargs` với tùy chọn `-n 1` hoặc `-L 1` để xử lý đầu ra được tạo bởi `find` có thể là không cần thiết. Lệnh `find` có tùy chọn `-exec` để chạy một lệnh nhất định cho mỗi một mục kết quả tìm kiếm.

Nếu đường dẫn có ký tự khoảng trắng, chúng ta phải chạy `find` với tùy chọn `-print0`. Tùy chọn này sẽ hướng dẫn `find` sử dụng một ký tự rỗng giữa mỗi mục nhập để danh sách có thể được phân tích cú pháp chính xác bởi `xargs` (đầu ra đã bị nén):

```
$ find . -name '*avi' -print0 -o -name '*mp4' -print0 -o -name '*mkv' -print0 | xargs -0 du
| sort -n
```

Tùy chọn `-0` đã cho `xargs` biết rằng ký tự rỗng nên được sử dụng làm dấu phân cách. Bằng cách đó, các đường dẫn tệp do `find` cung cấp sẽ được phân tích cú pháp chính xác ngay cả khi chúng có các ký tự trống hoặc ký tự đặc biệt khác trong đó. Ví dụ trước đã cho chúng ta thấy cách sử dụng lệnh `du` để tìm hiểu mức sử dụng đĩa của mọi tệp được tìm thấy và sau đó sắp xếp kết quả theo kích thước. Đầu ra đã bị nén cho ngắn gọn. Hãy lưu ý rằng, đối với mỗi tiêu chí tìm kiếm, ta phải đặt tùy chọn `-print0` cho `find`.

Theo mặc định, `xargs` sẽ đặt các đối số của lệnh được thực hiện sau cùng. Để thay đổi hành vi đó, ta phải sử dụng tùy chọn `-I`:

```
$ find . -mindepth 2 -name '*avi' -print0 -o -name '*mp4' -print0 -o -name '*mkv' -print0 |
xargs -0 -I PATH mv PATH ./
```

Trong ví dụ trước, mọi tệp được tìm thấy bởi `find` sẽ được chuyển đến thư mục hiện tại. Vì các đường dẫn nguồn phải được thông báo cho `mv` trước đường dẫn mục tiêu nên một thuật ngữ thay thế sẽ được cung cấp cho tùy chọn `-I` của `xargs` và sau đó được đặt một cách thích hợp bên cạnh `mv`. Bằng cách sử dụng ký tự rỗng làm dấu phân cách, ta sẽ không cần phải đặt thuật ngữ thay thế trong dấu trích dẫn kép.

Bài tập Hướng dẫn

1. Việc lưu ngày thực hiện các hành động được thực thi bởi các tệp lệnh tự động là rất tiện lợi. Lệnh `date +%Y-%m-%d` hiển thị ngày hiện tại ở định dạng *năm-tháng-ngày*. Làm thế nào để đầu ra của một lệnh như vậy có thể được lưu trữ trong một biến vỏ có tên là `TODAY` bằng cách sử dụng trình thay thế lệnh?

2. Bằng cách sử dụng lệnh `echo`, làm cách nào để gửi nội dung của biến `TODAY` đến đầu vào tiêu chuẩn của lệnh `sed s/-/./g`?

3. Làm cách nào để đầu ra của lệnh `date +%Y-%m-%d` được sử dụng làm Here string cho lệnh `sed s/-/./g`?

4. Lệnh `convert image.jpeg -resize 25% small/image.jpeg` tạo một phiên bản nhỏ hơn của `image.jpeg` và đặt hình ảnh thu được vào một tệp có tên tương tự bên trong thư mục con `small`. Bằng cách sử dụng `xargs`, làm thế nào để có thể thực hiện cùng một lệnh cho mọi hình ảnh được liệt kê trong tệp `filelist.txt`?

Bài tập Mở rộng

1. Một thói quen sao lưu đơn giản định kỳ sẽ tạo một hình ảnh của phân vùng `/dev/sda1` với lệnh `dd < /dev/sda1 > sda1.img`. Để thực hiện kiểm tra tính toàn vẹn của dữ liệu trong tương lai, tiến trình cũng sẽ tạo một hàm băm SHA1 của tệp với `sha1sum < sda1.img > sda1.sha1`. Bằng cách thêm đường ống và lệnh `tee`, hai lệnh này sẽ được kết hợp thành một như thế nào?

2. Lệnh `tar` được sử dụng để lưu trữ nhiều tệp vào một tệp duy nhất trong khi vẫn giữ nguyên cấu trúc thư mục. Tùy chọn `-T` cho phép chỉ định một tệp chứa các đường dẫn sẽ được lưu trữ. Ví dụ: `find /etc -type f | tar -cJ -f /srv/backup/etc.tar.xz -T -` sẽ tạo một tệp tar nén `etc.tar.xz` từ danh sách được cung cấp bởi lệnh `find` (tùy chọn `-T -` cho biết đầu vào tiêu chuẩn dưới dạng danh sách đường dẫn). Để tránh các lỗi phân tích cú pháp có thể xảy ra do đường dẫn có chứa dấu cách, ta nên thêm các tùy chọn lệnh nào cho `find` và `tar`?

3. Thay vì mở một phiên vỏ từ xa mới, lệnh `ssh` chỉ có thể thực thi một lệnh được chỉ định làm đối số của nó: `ssh user@storage "remote command"`. Giả sử rằng `ssh` cũng cho phép chuyển hướng đầu ra tiêu chuẩn của chương trình cục bộ sang đầu vào tiêu chuẩn của chương trình từ xa, làm cách nào để lệnh `cat` dẫn một tệp cục bộ có tên `etc.tar.gz` sang `/srv/backup/etc.tar.gz` tại `user@storage` thông qua `ssh`?

Tóm tắt

Bài học này bao gồm các kỹ thuật giao tiếp giữa các tiến trình truyền thống được sử dụng bởi Linux. *Dẫn ống Lệnh* sẽ tạo một phương thức giao tiếp một chiều giữa hai tiến trình và *trình thay thế lệnh* sẽ cho phép lưu trữ đầu ra của một tiến trình trong một biến vỏ. Bài học đã đi qua các bước sau:

- Cách *các đường ống* có thể được sử dụng để truyền đầu ra của một tiến trình sang đầu vào của một tiến trình khác.
- Mục đích của lệnh `tee` và `xargs`.
- Cách bắt đầu ra của một tiến trình với *trình thay thế lệnh*, lưu trữ nó trong một biến hoặc sử dụng nó trực tiếp như một tham số cho một lệnh khác.

Các lệnh và tiến trình đã được nhắc tới là:

- Dẫn ống lệnh bằng `|`.
- Thay thế lệnh bằng dấu trích dẫn ngược và `$()`.
- Các lệnh `tee`, `xargs` và `find`.

Đáp án Bài tập Hướng dẫn

1. Việc lưu ngày thực hiện các hành động được thực thi bởi các tệp lệnh tự động là rất tiện lợi. Lệnh `date +%Y-%m-%d` hiển thị ngày hiện tại ở định dạng *năm-tháng-ngày*. Làm thế nào để đầu ra của một lệnh như vậy có thể được lưu trữ trong một biến vỏ có tên là `TODAY` bằng cách sử dụng trình thay thế lệnh?

   ```
   $ TODAY=`date +%Y-%m-%d`
   ```

   hoặc

   ```
   $ TODAY=$(date +%Y-%m-%d)
   ```

2. Bằng cách sử dụng lệnh `echo`, làm cách nào để gửi nội dung của biến `TODAY` đến đầu vào tiêu chuẩn của lệnh `sed s/-/./g`?

   ```
   $ echo $TODAY | sed s/-/./g
   ```

3. Làm cách nào để đầu ra của lệnh `date +%Y-%m-%d` được sử dụng làm Here string cho lệnh `sed s/-/./g`?

   ```
   $ sed s/-/./g <<< `date +%Y-%m-%d`
   ```

   hoặc

   ```
   $ sed s/-/./g <<< $(date +%Y-%m-%d)
   ```

4. Lệnh `convert image.jpeg -resize 25% small/image.jpeg` tạo một phiên bản nhỏ hơn của `image.jpeg` và đặt hình ảnh thu được vào một tệp có tên tương tự bên trong thư mục con `small`. Bằng cách sử dụng `xargs`, làm thế nào để có thể thực hiện cùng một lệnh cho mọi hình ảnh được liệt kê trong tệp `filelist.txt`?

   ```
   $ xargs -I IMG convert IMG -resize 25% small/IMG < filelist.txt
   ```

   hoặc

```
$ cat filelist.txt | xargs -I IMG convert IMG -resize 25% small/IMG
```

Đáp án Bài tập Mở rộng

1. Một thói quen sao lưu đơn giản định kỳ sẽ tạo một hình ảnh của phân vùng `/dev/sda1` với lệnh `dd < /dev/sda1 > sda1.img`. Để thực hiện kiểm tra tính toàn vẹn của dữ liệu trong tương lai, tiến trình cũng tạo một hàm băm SHA1 của tệp với `sha1sum < sda1.img > sda1.sha1`. Bằng cách thêm đường ống và lệnh `tee`, hai lệnh này sẽ được kết hợp thành một như thế nào?

   ```
   # dd < /dev/sda1 | tee sda1.img | sha1sum > sda1.sha1
   ```

2. Lệnh `tar` được sử dụng để lưu trữ nhiều tệp vào một tệp duy nhất trong khi vẫn giữ nguyên cấu trúc thư mục. Tùy chọn `-T` cho phép chỉ định một tệp chứa các đường dẫn sẽ được lưu trữ. Ví dụ: `find /etc -type f | tar -cJ -f /srv/backup/etc.tar.xz -T -` sẽ tạo một tệp tar nén `etc.tar.xz` từ danh sách được cung cấp bởi lệnh `find` (tùy chọn `-T -` cho biết đầu vào tiêu chuẩn dưới dạng danh sách đường dẫn). Để tránh các lỗi phân tích cú pháp có thể xảy ra do đường dẫn có chứa dấu cách, ta nên thêm các tùy chọn lệnh nào cho `find` và `tar`?

   Các tuỳ chọn `-print0` và `--null`:

   ```
   $ find /etc -type f -print0 | tar -cJ -f /srv/backup/etc.tar.xz --null -T -
   ```

3. Thay vì mở một phiên vỏ từ xa mới, lệnh `ssh` chỉ có thể thực thi một lệnh được chỉ định làm đối số của nó: `ssh user@storage "remote command"`. Giả sử rằng `ssh` cũng cho phép chuyển hướng đầu ra tiêu chuẩn của chương trình cục bộ sang đầu vào tiêu chuẩn của chương trình từ xa, làm cách nào để lệnh `cat` dẫn một tệp cục bộ có tên `etc.tar.gz` sang `/srv/backup/etc.tar.gz` tại `user@storage` thông qua `ssh`?

   ```
   $ cat etc.tar.gz | ssh user@storage "cat > /srv/backup/etc.tar.gz"
   ```

   hoặc

   ```
   $ ssh user@storage "cat > /srv/backup/etc.tar.gz" < etc.tar.gz
   ```

103.5 Tạo, theo dõi và chấm dứt các Tiến trình

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 103.5

Khối lượng

4

Các lĩnh vực kiến thức chính

- Chạy các công việc nổi và ngầm.
- Báo hiệu chương trình tiếp tục chạy sau khi đăng xuất.
- Giám sát các tiến trình đang hoạt động.
- Chọn và sắp xếp các tiến trình để hiển thị.
- Gửi tín hiệu đến các tiến trình.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

- `&`
- `bg`
- `fg`
- `jobs`
- `kill`
- `nohup`
- `ps`
- `top`
- `free`

- `uptime`
- `pgrep`
- `pkill`
- `killall`
- `watch`
- `screen`
- `tmux`

103.5 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.1 Tạo, theo dõi và chấm dứt các Tiến trình
Bài:	1 trên 2

Giới thiệu

Mỗi khi chúng ta gọi một lệnh, một hoặc nhiều tiến trình sẽ được bắt đầu. Một quản trị viên hệ thống được đào tạo tốt không chỉ phải biết cách tạo các tiến trình mà còn phải theo dõi và gửi tới chúng các loại tín hiệu khác nhau nếu/khi cần. Trong bài học này, chúng ta sẽ xem xét về việc kiểm soát công việc và cách giám sát các tiến trình.

Kiểm soát Công việc

Công việc là các tiến trình đã được bắt đầu một cách tương tác thông qua cửa sổ dòng lệnh, sau đó chuyển sang chạy ngầm và chưa được thực thi xong. Chúng ta có thể xem các công việc đang hoạt động (và trạng thái của chúng) trong hệ thống Linux của mình bằng cách chạy lệnh `jobs`:

```
$ jobs
```

Lệnh `jobs` ở trên không tạo ra bất kỳ một kết quả nào. Điều này có nghĩa là không có công việc nào hiện đang hoạt động. Hãy cùng tạo một công việc đầu tiên bằng cách chạy một lệnh cần một khoảng thời gian nhất định để có thể hoàn thành (lệnh `sleep` với tham số `60`) và — trong khi

chạy — hãy nhấn `Ctrl` + `Z`:

```
$ sleep 60
^Z
[1]+  Stopped                 sleep 60
```

Việc thực thi lệnh đã bị dừng (hay đúng hơn là bị đình chỉ) và dấu nhắc lệnh đã khả dụng trở lại. Ta có thể tìm kiếm các công việc một lần nữa và sẽ tìm thấy công việc đã *bị đình chỉ* ngay lập tức:

```
$ jobs
[1]+  Stopped                 sleep 60
```

Hãy cùng giải thích đầu ra:

`[1]`

Số này là ID của công việc và có thể được sử dụng — đứng trước ký hiệu phần trăm (`%`) — để thay đổi trạng thái công việc bằng các tiện ích `fg`, `bg` và `kill` (như sẽ được đề cập ở phần sau).

`+`

Dấu cộng cho biết công việc mặc định hiện tại (tức công việc cuối cùng bị đình chỉ hoặc được chuyển thành chạy ngầm). Công việc trước đó được đánh dấu bằng dấu trừ (`-`). Bất kỳ công việc nào trước đó đều sẽ không được gắn cờ.

`Stopped`

Mô tả trạng thái công việc.

`sleep 60`

Bản thân lệnh hoặc công việc.

Với tùy chọn `-l`, các công việc sẽ hiển thị thêm ID của tiến trình (PID) ngay trước trạng thái:

```
$ jobs -l
[1]+  1114 Stopped                 sleep 60
```

Các tùy chọn có thể còn lại của công việc là:

`-n`

Chỉ liệt kê các tiến trình đã thay đổi trạng thái kể từ thông báo cuối cùng. Các trạng thái có thể bao gồm, `Running` (Đang chạy), `Stopped` (Đã dừng), `Terminated` (Đã chấm dứt) hoặc `Done`

(Hoàn thành).

`-p`

Liệt kê các ID của tiến trình.

`-r`

Chỉ liệt kê các công việc đang chạy.

`-s`

Chỉ liệt kê các công việc đã dừng (hoặc bị đình chỉ).

NOTE Hãy nhớ rằng, một công việc sẽ có cả *ID công việc* và *ID tiến trình* (PID).

Đặc tả Công việc

Lệnh `jobs`, cũng giống như các tiện ích khác như `fg`, `bg` và `kill` (sẽ được đề cập tới trong phần tiếp theo), đều cần một đặc tả công việc (hoặc `jobspec`) để thực hiện một công việc cụ thể. Như chúng ta vừa thấy, đây có thể (và thông thường) là ID công việc đứng trước `%`. Tuy nhiên, chúng ta cũng có thể có các đặc tả công việc khác nữa. Hãy cùng xem kỹ các đặc tả này:

`%n`

Công việc có số id là `n`:

```
$ jobs %1
[1]+ Stopped         sleep 60
```

`%str`

Công việc có dòng lệnh bắt đầu bằng `str`:

```
$ jobs %sl
[1]+ Stopped         sleep 60
```

`%?str`

Công việc có dòng lệnh có chứa `str`:

```
$ jobs %?le
[1]+ Stopped         sleep 60
```

`%+` hoặc `%%`

Công việc hiện tại (trong chế độ chạy nền hoặc bị đình chỉ trong danh sách chạy nổi):

```
$ jobs %+
[1]+ Stopped         sleep 60
```

`%-`

Công việc trước đây (công việc từng là `%+` trước công việc mặc định hiện tại):

```
$ jobs %-
[1]+ Stopped         sleep 60
```

Trong trường hợp của chúng ta, vì chỉ có một công việc nên nó vừa là công việc hiện tại vừa là công việc trước đó.

Trạng thái công việc: Đình chỉ, Chạy nổi và Chạy ngầm

Một khi công việc ở chế độ chạy ngầm hoặc đã bị đình chỉ, chúng ta có thể thực hiện bất kỳ điều nào trong số ba điều sau:

1. Đưa nó lên chạy nổi với `fg`:

   ```
   $ fg %1
   sleep 60
   ```

   `fg` sẽ chuyển công việc đã chỉ định thành chạy nổi và biến nó thành công việc hiện tại. Bây giờ, chúng ta có thể đợi cho đến khi nó kết thúc và dừng lại bằng `Ctrl` + `Z` hoặc chấm dứt bằng `Ctrl` + `C`.

2. Đưa nó vào chạy ngầm với `bg`:

   ```
   $ bg %1
   [1]+ sleep 60 &
   ```

   Khi chạy ngầm, công việc có thể được đưa trở lại chạy nổi bằng `fg` hoặc bị chấm dứt (xem bên dưới). Hãy lưu ý ký tự (&) có nghĩa là công việc đã được chuyển thành chạy ngầm. Trên thực tế, ta cũng có thể sử dụng ký tự (&) để bắt đầu một tiến trình trực tiếp ở chế độ chạy ngầm:

   ```
   $ sleep 100 &
   ```

```
[2] 970
```

Cùng với ID công việc của công việc mới (`[2]`), giờ đây, chúng ta cũng sẽ nhận được cả ID tiến trình của nó (`970`). Bây giờ, cả hai công việc đều đang chạy ngầm:

```
$ jobs
[1]- Running         sleep 60 &
[2]+ Running         sleep 100 &
```

Một lát sau, công việc đầu tiên sẽ kết thúc việc thực thi:

```
$ jobs
[1]- Done          sleep 60
[2]+ Running         sleep 100 &
```

3. Chấm dứt nó thông qua tín hiệu `SIGTERM` với `kill`:

```
$ kill %2
```

Để đảm bảo rằng công việc đã bị chấm dứt, hãy chạy lại `jobs`:

```
$ jobs
[2]+ Terminated         sleep 100
```

NOTE Nếu không có công việc nào được chỉ định, `fg` và `bg` sẽ thực thi với công việc mặc định hiện tại. Tuy nhiên, `kill` luôn cần có một đặc tả công việc.

Công việc tách rời: `nohup`

Tất cả các công việc mà chúng ta đã thấy trong các phần trước đều được gắn với phiên của người dùng đã gọi chúng. Điều đó có nghĩa là nếu phiên kết thúc, công việc cũng sẽ biến mất. Tuy nhiên, ta có thể tách các công việc khỏi phiên và chạy chúng ngay cả sau khi phiên đã kết thúc. Điều này có thể được thực hiện bằng lệnh `nohup` (“no hangup”). Cú pháp của lệnh sẽ như sau:

```
nohup COMMAND &
```

Hãy nhớ rằng, `&` sẽ chuyển tiến trình sang chạy ngầm và giải phóng cửa sổ dòng lệnh mà chúng ta đang sử dụng.

Hãy cùng tách công việc đang chạy ngầm `ping localhost` ra khỏi phiên hiện tại:

```
$ nohup ping localhost &
[1] 1251
$ nohup: ignoring input and appending output to 'nohup.out'
^C
```

Đầu ra đã hiển thị cho chúng ta thấy ID công việc (`[1]`) và PID (`1251`), theo sau là một thông báo cho biết về tệp `nohup.out`. Đây là tệp mặc định nơi `stdout` và `stderr` sẽ được lưu. Bây giờ, chúng ta có thể nhấn `Ctrl` + `C` để giải phóng dấu nhắc lệnh, đóng phiên, bắt đầu một phiên khác với tên `root` và sử dụng `tail -f` để kiểm tra xem lệnh có đang chạy không cũng như đầu ra có được ghi vào tệp mặc định hay không:

```
$ 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 Thay vì sử dụng `nohup.out` mặc định, bạn có thể chỉ định tệp đầu ra mà bạn chọn với `nohup ping localhost > /path/to/your/file &`.

Nếu muốn chấm dứt tiến trình, chúng ta sẽ phải chỉ định PID của nó:

```
# kill 1251
```

Giám sát Tiến trình

Một tiến trình hoặc tác vụ đều là bản thể của một chương trình đang chạy. Điều này có nghĩa là chúng ta sẽ tạo ra các tiến trình mới mỗi khi nhập lệnh vào cửa sổ dòng lệnh.

Lệnh `watch` sẽ thực thi chương trình theo định kỳ (mỗi 2 giây theo mặc định) và sẽ cho phép chúng ta *theo dõi* những thay đổi ở đầu ra của chương trình theo thời gian. Chẳng hạn, chúng ta có thể giám sát mức trung bình tải thay đổi như thế nào khi có nhiều tiến trình được chạy bằng cách nhập `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
```

Lệnh sẽ chạy cho đến khi bị gián đoạn. Vì vậy, chúng ta sẽ phải dừng lệnh bằng `Ctrl` + `C`. Ta sẽ nhận được hai dòng dưới dạng đầu ra: dòng đầu tiên tương ứng với `watch` và cho biết tần suất lệnh sẽ được chạy (`Every 2.0s: uptime`), lệnh/chương trình nào cần xem (`uptime`) cũng như tên máy chủ và thời gian (`debian: Tue Aug 20 23:31:27 2019`). Dòng thứ hai cho thấy thời gian hoạt động và trong đó bao gồm thời gian (`23:31:27`), thời gian hệ thống đã hoạt động (`up 21 min`), số lượng người dùng đang hoạt động (`1 user`) và tải lượng trung bình của hệ thống/ số lượng tiến trình đang thực thi hoặc ở trạng thái chờ trong 1, 5 và 15 phút qua (`load average: 0,00, 0,00, 0,00`).

Tương tự, ta có thể kiểm tra việc sử dụng bộ nhớ khi các tiến trình mới được tạo bằng lệnh `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
```

Để thay đổi khoảng thời gian cập nhật cho `watch`, hãy sử dụng các tùy chọn `-n` hoặc `--interval` cộng với số giây như dưới đây:

```
$ watch -n 5 free
```

Bây giờ, lệnh `free` sẽ chạy sau mỗi 5 giây.

Để biết thêm thông tin về các tùy chọn cho `uptime`, `free` và `watch`, hãy tham khảo các trang hướng dẫn của chúng.

NOTE Thông tin do `uptime` và `free` cung cấp cũng được tích hợp trong các công cụ toàn diện hơn là `top` và `ps` (xem bên dưới).

Gửi tín hiệu tới các Tiến trình: `kill`

Mỗi tiến trình đơn lẻ đều có một mã định danh tiến trình hoặc một PID duy nhất. Cách để tìm ra PID của một tiến trình là sử dụng lệnh `pgrep`, theo sau là tên của tiến trình:

```
$ pgrep sleep
1201
```

NOTE Mã định danh tiến trình cũng có thể được biết thông qua lệnh `pidof` (ví dụ: `pidof sleep`).

Tương tự như `pgrep`, lệnh `pkill` sẽ chấm dứt một tiến trình dựa trên tên của nó:

```
$ pkill sleep
[1]+  Terminated              sleep 60
```

Để chấm dứt nhiều phiên bản của cùng một tiến trình, ta có thể sử dụng lệnh `killall`:

```
$ sleep 60 &
[1] 1246
$ sleep 70 &
[2] 1247
$ killall sleep
[1]-  Terminated              sleep 60
[2]+  Terminated              sleep 70
```

Cả `pkill` và `killall` đều hoạt động giống như `kill` ở chỗ chúng sẽ gửi tín hiệu kết thúc tới (các) tiến trình đã chỉ định. Nếu không có tín hiệu nào được cung cấp, tín hiệu mặc định của `SIGTERM` sẽ được gửi. Tuy nhiên, `kill` sẽ chỉ lấy một công việc hoặc một ID tiến trình làm đối số.

Tín hiệu có thể được chỉ định bởi:

- Tên:

  ```
  $ kill -SIGHUP 1247
  ```

- Số:

  ```
  $ kill -1 1247
  ```

- Tuỳ chọn:

  ```
  $ kill -s SIGHUP 1247
  ```

Để `kill` hoạt động theo cách tương tự như `pkill` hoặc `killall` (và tự lưu các lệnh để tìm ra PID trước), chúng ta có thể sử dụng trình thay thế lệnh:

```
$ kill -1 $(pgrep sleep)
```

Như chúng ta đã biết, một cú pháp thay thế sẽ là `kill -1 `pgrep sleep``.

TIP Để có danh sách đầy đủ tất cả các tín hiệu `kill` và mã của chúng, hãy nhập `kill -l` vào cửa sổ dòng lệnh. Hãy sử dụng `-KILL` (`-9` hoặc `-s KILL`) để chấm dứt các tiến trình nổi loạn khi bất kỳ tín hiệu nào khác bị lỗi.

`top` và `ps`

Khi nói đến giám sát tiến trình, hai công cụ rất có giá trị là `top` và `ps`. Trong khi `top` tạo ra đầu ra động thì `ps` lại tạo đầu ra tĩnh. Trong mọi trường hợp, cả hai đều là những tiện ích tuyệt vời để chúng ta có cái nhìn toàn diện về tất cả các tiến trình trong hệ thống.

Tương tác với `top`

Để gọi `top`, ta chỉ cần gõ `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
```

```
    ( . . . )
```

`top` cho phép người dùng có một số tương tác. Theo mặc định, đầu ra sẽ được sắp xếp theo tỷ lệ phần trăm thời gian CPU được sử dụng bởi mỗi tiến trình theo thứ tự giảm dần. Hành vi này có thể được sửa đổi bằng cách nhấn các phím sau từ bên trong `top`:

`M`

Sắp xếp theo số lượng *bộ nhớ* được sử dụng.

`N`

Sắp xếp theo *số* ID tiến trình.

`T`

Sắp xếp theo *thời gian*.

`P`

Sắp xếp theo *phần trăm* sử dụng CPU.

TIP Để chuyển đổi giữa thứ tự giảm dần/tăng dần, chỉ cần nhấn `R`.

Các tuỳ chọn thú vị khác để tương tác với `top` là:

`?` hoặc `h`

Cần giúp đỡ.

`k`

Chấm dứt một tiến trình. `top` sẽ yêu cầu huỷ `PID` của tiến trình cũng như yêu cầu gửi tín hiệu (theo mặc định là `SIGTERM` hoặc `15`).

`r`

Thay đổi mức độ ưu tiên của một tiến trình (`renice`). `top` sẽ hỏi bạn về giá trị `nice`. Các giá trị có thể nằm trong khoảng từ -20 đến 19, nhưng chỉ siêu người dùng (`root`) mới có thể đặt giá trị đó thành âm hoặc thấp hơn giá trị hiện tại.

`u`

Liệt kê các tiến trình từ một người dùng cụ thể (theo mặc định, các tiến trình từ tất cả mọi người dùng đều sẽ được hiển thị).

`c`

Hiển thị đường dẫn tuyệt đối của chương trình và phân biệt giữa tiến trình dung lượng người dùng và tiến trình dung lượng hạt nhân (trong dấu ngoặc vuông).

`V`

Chế độ xem cây/thứ bậc tiến trình

`t` và `m`

Thay đổi giao diện của các lần đọc CPU và bộ nhớ tương ứng theo chu kỳ bốn giai đoạn: hai lần nhấn đầu tiên hiển thị thanh tiến trình, lần nhấn thứ ba ẩn thanh tiến trình và lần nhấn thứ tư đưa thanh tiến trình trở lại.

`W`

Lưu cài đặt cấu hình vào `~/.toprc`.

TIP

Một phiên bản cao cấp và thân thiện hơn với người dùng của `top` là `htop`. Một cách khác — có lẽ toàn diện hơn — là `atop`. Nếu chưa được cài đặt trong hệ thống, hãy sử dụng trình quản lý gói của bạn để cài đặt chúng và dùng thử.

Giải thích về đầu ra của `top`

Đầu ra của `top` được chia thành hai vùng: vùng *tóm tắt* và vùng *tác vụ*.

Vùng tóm tắt trong `top`

Vùng tóm tắt được tạo thành từ năm hàng trên cùng và sẽ cung cấp cho chúng ta các thông tin sau:

- `top - 11:10:29 up 2:21, 1 user, load average: 0,11, 0,20, 0,14`
  - thời gian hiện tại (ở định dạng 24 giờ): `11:20:29`
  - thời gian hoạt động (thời gian hệ thống đã hoạt động và chạy): `up 2:21`
  - số người dùng đăng nhập và tải lượng trung bình của CPU trong 1, 5 và 15 phút gần nhất tương ứng: `load average: 0,11, 0,20, 0,14`
- `Tasks: 73 total, 1 running, 72 sleeping, 0 stopped, 0 zombie` (thông tin về các tiến trình)
  - tổng số tiến trình ở chế độ hoạt động: `73 total`
  - đang chạy (những tiến trình đang được thực thi): `1 running`
  - đang ngủ (những tiến trình đang chờ để tiếp tục thực thi): `72 sleeping`
  - đã dừng (bằng tín hiệu kiểm soát công việc): `0 stopped`
  - zombie (những tiến trình đã hoàn thành thực thi nhưng vẫn đang đợi tiến trình mẹ xóa chúng khỏi bảng tiến trình): `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`

(phần trăm CPU tiêu thụ)

- tiến trình người dùng: `0,0 us`
- tiến trình hệ thống/hạt nhân: `0,4 sy`
- tiến trình được đặt thành giá trị *chuẩn* — giá trị càng chuẩn, mức độ ưu tiên càng thấp: `0,0 ni`
- không có gì — thời gian CPU nhàn rỗi: `99,7 id`
- các tiến trình chờ thao tác I/O: `0,0 wa`
- tiến trình phục vụ ngắt phần cứng — thiết bị ngoại vi gửi tín hiệu mà bộ xử lý cần lưu ý: `0,0 hi`
- tiến trình phục vụ ngắt phần mềm: `0,0 si`
- các tiến trình phục vụ các tác vụ của máy ảo khác trong môi trường ảo từ đó có thời gian đánh cắp (?): `0,0 st`

- `KiB Mem : 1020332 total, 909492 free, 38796 used, 72044 buff/cache` (memory information in kilobytes)
  - tổng dung lượng bộ nhớ: `1020332 total`
  - bộ nhớ chưa sử dụng: `909492 miễn phí`
  - bộ nhớ đang sử dụng: `38796 used`
  - bộ nhớ được đệm và lưu vào bộ nhớ đệm để tránh truy cập đĩa quá nhiều: `72044 buff/cache`

    Hãy lưu ý `total` là tổng của ba giá trị còn lại — `free`, `used` và `buff/cache` — (khoảng 1 GB trong trường hợp của chúng ta).

- `KiB Swap: 1046524 total, 1046524 free, 0 used. 873264 avail Mem` (thông tin hoán đổi tính theo kilobyte)
  - tổng dung lượng hoán đổi: `1046524 total`
  - dung lượng hoán đổi chưa sử dụng: `1046524 miễn phí`
  - dung lượng hoán đổi đang sử dụng: `0 used`
  - lượng bộ nhớ hoán đổi có thể được phân bổ cho các tiến trình mà không gây ra thêm phiên hoán đổi: `873264 avail Mem`

Khu vực Nhiệm vụ trong `top`: Trường và Cột

Bên dưới *vùng tóm tắt* có *khu vực tác vụ* bao gồm một loạt các *trường* và *cột* để báo cáo thông tin

về các tiến trình đang chạy:

PID

Định danh tiến trình.

USER

Người dùng đã đưa ra lệnh tạo tiến trình.

PR

Ưu tiên của quá trình cho hạt nhân.

NI

Giá trị nice của quá trình. Giá trị nice thấp hơn sẽ có mức độ ưu tiên cao hơn.

VIRAT

Tổng dung lượng bộ nhớ được sử dụng bởi tiến trình (bao gồm cả Hoán đổi).

RES

Bộ nhớ RAM được sử dụng bởi tiến trình.

SHR

Bộ nhớ dùng chung của tiến trình với các tiến trình khác.

S

Trạng thái của tiến trình. Các giá trị bao gồm: `S` (ngủ gián đoạn — chờ một sự kiện kết thúc), `R` (có thể chạy được — đang thực thi hoặc đang xếp hàng chờ được thực thi) hoặc `Z` (zombie — các tiến trình con bị kết thúc có cấu trúc dữ liệu chưa được xóa khỏi bảng tiến trình).

%CPU

Phần trăm CPU được sử dụng bởi tiến trình.

%MEM

Tỷ lệ phần trăm RAM được sử dụng bởi tiến trình, tức giá trị `RES` được biểu thị dưới dạng phần trăm.

TIME+

Tổng thời gian hoạt động của tiến trình.

COMMAND

Tên của lệnh/chương trình đã tạo ra tiến trình.

Xem các Tiến trình tĩnh: `ps`

Như đã nói ở trên, `ps` sẽ hiển thị hình ảnh tức thời của các tiến trình. Để xem tất cả các tiến trình với một cửa sổ dòng lệnh (tty), hãy nhập `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
 (...)
```

Lý giải cú pháp và đầu ra của tuỳ chọn `ps`

Liên quan đến các tùy chọn, `ps` có thể chấp nhận ba kiểu khác nhau: BSD, UNIX và GNU. Hãy cùng xem mỗi kiểu này sẽ hoạt động như thế nào khi báo cáo thông tin về một ID tiến trình cụ thể:

BSD

Các tùy chọn không theo sau bất kỳ dấu gạch ngang nào:

```
$ ps p 811
 PID TTY   STAT  TIME COMMAND
 811 pts/0  S   0:00 -su
```

UNIX

Các tùy chọn theo sau một dấu gạch ngang:

```
$ ps -p 811
 PID TTY     TIME CMD
 811 pts/0  00:00:00 bash
```

GNU

Các tùy chọn được theo sau bởi hai dấu gạch ngang:

```
$ ps --pid 811
 PID TTY     TIME CMD
 811 pts/0  00:00:00 bash
```

Trong cả ba trường hợp, `ps` sẽ báo cáo thông tin về tiến trình có `PID` là `811` — trong trường hợp này là `bash`.

Tương tự, ta có thể sử dụng `ps` để tìm kiếm các tiến trình được bắt đầu bởi một người dùng cụ thể:

- `ps U carol` (BSD)
- `ps -u carol` (UNIX)
- `ps --user carol` (GNU)

Hãy cùng kiểm tra các tiến trình được bắt đầu bởi `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
```

Carol đã bắt đầu hai tiến trình: `bash` (`-su`) và `ps` (`ps U carol`). Cột `STAT` cho chúng ta biết trạng thái của tiến trình (xem bên dưới).

Chúng ta có thể tận dụng tối đa `ps` bằng cách kết hợp một số tùy chọn của nó. Một lệnh rất hữu ích (tạo đầu ra tương tự như đầu ra của `top`) là `ps aux` (kiểu BSD). Trong trường hợp này, các tiến trình từ tất cả các vỏ (không chỉ vỏ hiện tại) sẽ được hiển thị. Ý nghĩa của các khoá chuyển sẽ như sau:

a
: Hiển thị các tiến trình được đính kèm với `tty` hoặc cửa sổ dòng lệnh.

u
: Hiển thị định dạng hướng về người dùng.

x
: Hiển thị các tiến trình không được đính kèm với `tty` hoặc cửa sổ dòng lệnh.

```
$ 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]
(...)
```

Hãy cùng giải thích các cột:

`USER`

Chủ sở hữu của quá trình.

`PID`

Định danh tiến trình.

`%CPU`

Tỷ lệ CPU được sử dụng.

`%MEM`

Tỷ lệ bộ nhớ vật lý được sử dụng.

`VSZ`

Bộ nhớ ảo của tiến trình tính theo KiB.

`RSS`

Bộ nhớ vật lý không hoán đổi được sử dụng bởi tiến trình trong KiB.

`TT`

Cửa sổ dòng lệnh (`tty`) kiểm soát tiến trình.

`STAT`

Mã đại diện cho trạng thái của tiến trình. Ngoài `S`, `R` và `Z` (mà chúng ta đã thấy khi mô tả đầu ra của `top`), các giá trị có thể khác bao gồm: `D` (ngủ liên tục — thường là chờ I/O) và `T` (dừng — thường là do tín hiệu điều khiển). Một số công cụ sửa đổi bổ sung bao gồm: `<` (mức độ ưu tiên cao — không "nice" với các tiến trình khác), `N` (mức độ ưu tiên thấp — "nice" với các tiến trình khác) hoặc `+` (trong nhóm tiến trình chạy nổi).

`STARTED`

Thời gian tiến trình bắt đầu.

`TIME`

Thời gian CPU tích lũy.

COMMAND

Lệnh bắt đầu tiến trình.

Bài tập Hướng dẫn

1. `oneko` là một chương trình vui nhộn hiển thị một chú mèo đuổi theo con trỏ chuột của bạn. Nếu chưa được cài đặt trong hệ thống, hãy cài đặt nó bằng trình quản lý gói của bản phân phối. Chúng ta sẽ sử dụng nó để hiểu thêm về tác vụ kiểm soát công việc.
   - Hãy khởi động chương trình. Làm thế nào để bạn làm được điều này?
   - Hãy di chuyển con trỏ chuột để xem chú mèo đuổi theo nó như thế nào. Sau đó, hãy đình chỉ tiến trình. Làm thế nào để bạn làm được điều này? Đầu ra sẽ là gì?
   - Hãy kiểm tra xem bạn hiện đang có bao nhiêu công việc. Bạn sẽ gõ gì? Đầu ra sẽ là gì?
   - Bây giờ, hãy chuyển nó thành một tiến trình chạy ngầm và chỉ định ID công việc của nó. Đầu ra sẽ là gì? Làm thế nào để bạn có thể biết được công việc đang chạy ngầm?
   - Cuối cùng, hãy kết thúc công việc bằng cách chỉ định ID công việc của nó. Bạn sẽ gõ gì?
2. Hãy khám phá PID của tất cả các tiến trình do máy chủ web *Apache HTTPD* (`apache2`) tạo ra bằng hai lệnh khác nhau.
3. Hãy chấm dứt tất cả các tiến trình `apache2` mà không sử dụng PID của chúng bằng hai lệnh khác nhau.
4. Giả sử bạn phải chấm dứt tất cả các bản thể của `apache2` và bạn không có thời gian để tìm hiểu xem PID của chúng là gì. Bạn sẽ thực hiện điều đó như thế nào bằng cách sử dụng `kill` với tín hiệu `SIGTERM` mặc định trong một dòng mã?
5. Hãy bắt đầu `top` và tương tác với nó bằng cách thực hiện như sau:

- Hiển thị chế độ xem cây tiến trình.

- Hiển thị đường dẫn đầy đủ của các tiến trình phân biệt giữa dung lượng người dùng và dung lượng hạt nhân.

6. Hãy nhập lệnh `ps` để hiển thị tất cả các tiến trình được bắt đầu bởi người dùng *Apache HTTPD web server* (`www-data`):
   - Sử dụng cú pháp BSD:

   - Sử dụng cú pháp UNIX:

   - Sử dụng cú pháp GNU:

Bài tập Mở rộng

1. Tín hiệu `SIGHUP` có thể được sử dụng như một cách để khởi động lại một số trình nền nhất định. Với máy chủ web *Apache HTTPD* — ví dụ — việc gửi `SIGHUP` tới tiến trình mẹ (tiến trình được bắt đầu bởi `init`) sẽ chấm dứt tiến trình con của nó. Tuy nhiên, tiến trình mẹ sẽ đọc lại các tệp cấu hình của nó, mở lại các tệp nhật ký và tạo ra một nhóm con mới. Hãy hực hiện các tác vụ sau:
   - Khởi động máy chủ web.
   - Đảm bảo rằng bạn biết PID của tiến trình mẹ.
   - Làm cho máy chủ web Apache HTTPD khởi động lại bằng cách gửi tín hiệu `SIGHUP` tới tiến trình mẹ.
   - Kiểm tra và đảm bảo rằng tiến trình mẹ không bị chấm dứt và những tiến trình con mới đã được tạo ra:
2. Mặc dù ban đầu là tĩnh nhưng đầu ra của `ps` có thể *biến* thành động bằng cách kết hợp `ps` và `watch`. Chúng ta sẽ giám sát máy chủ web *Apache HTTPD* cho các kết nối mới. Trước khi thực hiện các tác vụ được mô tả bên dưới, bạn nên đọc mô tả về chỉ thị `MaxConnectionsPerChild` trong Chỉ thị chung về MPM của Apache.
   - Hãy thêm chỉ thị `MaxConnectionsPerChild` với giá trị `1` trong tệp cấu hình của `apache2` — trong *Debian* và các dẫn xuất tại `/etc/apache2/apache2.conf`; trong họ *CentOS* là tại `/etc/httpd/conf/httpd.conf`. Đừng quên khởi động lại `apache2` để các thay đổi có hiệu lực.
   - Hãy nhập lệnh sử dụng `watch`, `ps` và `grep` cho các kết nối `apache2`.
   - Bây giờ, hãy mở trình duyệt web hoặc sử dụng trình duyệt dòng lệnh (chẳng hạn như `lynx`) để thiết lập kết nối với máy chủ web thông qua địa chỉ IP của nó. Bạn sẽ thấy gì trong đầu ra của `watch`?

3. Như chúng ta đã biết, theo mặc định, `top` sẽ sắp xếp các tác vụ dựa vào phần trăm mức sử dụng CPU theo thứ tự giảm dần (giá trị cao hơn ở trên cùng). Hành vi này có thể được sửa đổi bằng các phím tương tác `M` (mức sử dụng bộ nhớ), `N` (mã định danh duy nhất của tiến trình), `T` (thời gian chạy) và `P` (phần trăm thời gian của CPU). Tuy nhiên, bạn cũng có thể sắp xếp danh sách tác vụ theo ý thích của mình bằng cách khởi chạy `top` với khóa chuyển `-o` (để biết thêm thông tin, hãy xem trang hướng dẫn của top). Bây giờ, hãy thực hiện các tác vụ sau:
   - Khởi chạy `top` để các tác vụ được sắp xếp theo mức sử dụng bộ nhớ:
   - Xác minh rằng bạn đã gõ đúng lệnh bằng cách đánh dấu nổi cột bộ nhớ:
4. `ps` cũng có một khoá chuyển `o` để chỉ định các cột bạn muốn hiển thị. Hãy kiểm tra tùy chọn này và thực hiện các tác vụ sau:
   - Khởi chạy `ps` để chỉ hiển thị thông tin về *người dùng*, *phần trăm bộ nhớ đã sử dụng*, *phần trăm thời gian CPU đã sử dụng* và *lệnh đầy đủ*:
   - Bây giờ, hãy khởi chạy `ps` để thông tin duy nhất được hiển thị là thông tin về người dùng và tên của chương trình họ đang sử dụng:

Tóm tắt

Trong bài học này, chúng ta đã học về *công việc* và *kiểm soát công việc*. Các định nghĩa và khái niệm quan trọng cần ghi nhớ là:

- Công việc là các tiến trình được chạy ngầm.
- Ngoài *ID tiến trình*, các công việc cũng được chỉ định *ID công việc* khi được tạo.
- Để kiểm soát công việc, ta cần có đặc tả công việc (`jobspec`).
- Công việc có thể được đưa lên chạy nổi, chạy ngầm, đình chỉ và chấm dứt (hoặc *kết thúc*).
- Một công việc có thể được tách ra khỏi cửa sổ dòng lệnh và phiên mà nó được tạo.

Tương tự như vậy, chúng ta cũng đã thảo luận về khái niệm của *tiến trình* và *giám sát tiến trình*. Các ý quan trọng nhất là:

- Các tiến trình đang chạy chương trình.
- Các tiến trình có thể được giám sát.
- Các tiện ích khác nhau cho phép ta tìm ra *ID tiến trình* của các tiến trình cũng như gửi cho chúng các tín hiệu để chấm dứt.
- Tín hiệu có thể được chỉ định theo tên (ví dụ: `-SIGTERM`), số (ví dụ: `-15`) hoặc tùy chọn (ví dụ: `-s SIGTERM`).
- `top` và `ps` rất hiệu quả trong quá trình giám sát. Đầu ra của `top` là đầu ra động và liên tục cập nhật; đầu ra của `ps` lại là đầu ra tĩnh.

Các lệnh đã được dùng trong bài học:

`jobs`
: Hiển thị các công việc đang hoạt động và trạng thái của chúng.

`sleep`
: Trì hoãn trong một khoảng thời gian cụ thể.

`fg`
: Đưa công việc lên chạy nổi.

`bg`
: Đưa công việc xuống thành chạy ngầm.

`kill`

Chấm dứt công việc.

`nohup`

Tách công việc khỏi phiên/cửa sổ dòng lệnh.

`exit`

Thoát vỏ hiện tại.

`tail`

Hiển thị các dòng gần đây nhất trong một tệp.

`watch`

Chạy lặp lại một lệnh (chu kỳ 2 giây theo mặc định).

`uptime`

Hiển thị thời gian hệ thống đã chạy, số lượng người dùng hiện tại và mức tải lượng trung bình của hệ thống.

`free`

Hiển thị tình trạng sử dụng bộ nhớ.

`pgrep`

Tra cứu ID tiến trình dựa trên tên.

`pidof`

Tra cứu ID tiến trình dựa trên tên.

`pkill`

Gửi tín hiệu để xử lý theo tên.

`killall`

Kết thúc (các) tiến trình theo tên.

`top`

Hiển thị các tiến trình Linux.

`ps`

Báo cáo một bản sao hình ảnh tức thời của các tiến trình hiện tại.

Đáp án Bài tập Hướng dẫn

1. `oneko` là một chương trình vui nhộn hiển thị một chú mèo đuổi theo con trỏ chuột của bạn. Nếu chưa được cài đặt trong hệ thống, hãy cài đặt nó bằng trình quản lý gói của bản phân phối. Chúng ta sẽ sử dụng nó để hiểu thêm về tác vụ kiểm soát công việc.
   - Hãy khởi động chương trình. Làm thế nào để bạn làm được điều này?

     Bằng cách gõ `oneko` vào cửa sổ dòng lệnh.

   - Hãy di chuyển con trỏ chuột để xem chú mèo đuổi theo nó như thế nào. Sau đó, hãy đình chỉ tiến trình. Làm thế nào để bạn làm được điều này? Đầu ra sẽ là gì?

     Bằng cách nhấn tổ hợp phím `Ctrl` + `z`:

     ```
     [1]+ Stopped          oneko
     ```

   - Hãy kiểm tra xem bạn hiện đang có bao nhiêu công việc. Bạn sẽ gõ gì? Đầu ra sẽ là gì?

     ```
     $ jobs
     [1]+ Stopped          oneko
     ```

   - Bây giờ, hãy chuyển nó thành một tiến trình chạy ngầm và chỉ định ID công việc của nó. Đầu ra sẽ là gì? Làm thế nào để bạn có thể biết được công việc đang chạy ngầm?

     ```
     $ bg %1
     [1]+ oneko &
     ```

   - Cuối cùng, hãy kết thúc công việc chỉ định ID công việc của nó. Bạn sẽ gõ gì?

     +

```
$ *kill %1*
```

1. Hãy khám phá PID của tất cả các tiến trình do máy chủ web *Apache HTTPD* (`apache2`) tạo ra bằng hai lệnh khác nhau.

   ```
   $ pgrep apache2
   ```

hoặc

```
$ pidof apache2
```

2. Hãy chấm dứt tất cả các tiến trình `apache2` mà không sử dụng PID của chúng bằng hai lệnh khác nhau.

```
$ pkill apache2
```

hoặc

```
$ killall apache2
```

3. Giả sử bạn phải chấm dứt tất cả các bản thể của `apache2` và bạn không có thời gian để tìm hiểu xem PID của chúng là gì. Bạn sẽ thực hiện điều đó như thế nào bằng cách sử dụng `kill` với tín hiệu `SIGTERM` mặc định trong một dòng mã?

```
$ kill $(pgrep apache2)
$ kill `pgrep apache2`
```

hoặc

```
$ kill $(pidof apache2)
$ kill `pidof apache2`
```

NOTE Vì `SIGTERM` (`15`) là tín hiệu mặc định nên ta không cần truyền bất kỳ tùy chọn nào cho `kill`.

4. Hãy bắt đầu `top` và tương tác với nó bằng cách thực hiện như sau:
   - Hiển thị chế độ xem cây tiến trình.

     Nhấn `V`.

   - Hiển thị đường dẫn đầy đủ của các tiến trình phân biệt giữa dung lượng người dùng và dung lượng hạt nhân.

     Nhấn `c`.

5. Hãy nhập lệnh `ps` để hiển thị tất cả các tiến trình được bắt đầu bởi người dùng *Apache HTTPD web server* (`www-data`):
   - Sử dụng cú pháp BSD:

     ```
     $ ps U www-data
     ```

   - Sử dụng cú pháp UNIX:

     ```
     $ ps -u www-data
     ```

   - Sử dụng cú pháp GNU:

     ```
     $ ps --user www-data
     ```

Đáp án Bài tập Mở rộng

1. Tín hiệu `SIGHUP` có thể được sử dụng như một cách để khởi động lại một số trình nền nhất định. Với máy chủ web *Apache HTTPD* — ví dụ — việc gửi `SIGHUP` tới tiến trình mẹ (tiến trình được bắt đầu bởi `init`) sẽ chấm dứt tiến trình con của nó. Tuy nhiên, tiến trình mẹ sẽ đọc lại các tệp cấu hình của nó, mở lại các tệp nhật ký và tạo ra một nhóm con mới. Hãy hực hiện các tác vụ sau:
   - Khởi động máy chủ web.

     ```
     $ sudo systemctl start apache2
     ```

   - Đảm bảo rằng bạn biết PID của tiến trình mẹ.

     ```
     $ ps aux | grep apache2
     ```

     Tiến trình mẹ được bắt đầu bởi người dùng `root`. Trong trường hợp của chúng ta là tiến trình có PID `1653`.

   - Làm cho máy chủ web Apache HTTPD khởi động lại bằng cách gửi tín hiệu `SIGHUP` tới tiến trình mẹ.

     ```
     $ kill -SIGHUP 1653
     ```

   - Kiểm tra và đảm bảo rằng tiến trình mẹ không bị chấm dứt và những tiến trình con mới đã được tạo ra:

     ```
     $ ps aux | grep apache2
     ```

     Bây giờ, bạn sẽ thấy tiến trình `apache2` gốc cùng với hai tiến trình con mới.

2. Mặc dù ban đầu là tĩnh nhưng đầu ra của `ps` có thể *biến* thành động bằng cách kết hợp `ps` và `watch`. Chúng ta sẽ giám sát máy chủ web *Apache HTTPD* cho các kết nối mới. Trước khi thực hiện các tác vụ được mô tả bên dưới, bạn nên đọc mô tả về chỉ thị `MaxConnectionsPerChild` trong Chỉ thị chung về MPM của Apache.
   - Hãy thêm chỉ thị `MaxConnectionsPerChild` với giá trị `1` trong tệp cấu hình của `apache2` — trong *Debian* và các dẫn xuất tại `/etc/apache2/apache2.conf`; trong họ *CentOS* là tại `/etc/httpd/conf/httpd.conf`. Đừng quên khởi động lại `apache2` để các thay đổi

có hiệu lực.

Dòng cần có trong tệp cấu hình là `MaxConnectionsPerChild 1`. Một cách để khởi động lại máy chủ web là thông qua `sudo systemctl restart apache2`.

- Nhập lệnh sử dụng `watch`, `ps` và `grep` cho các kết nối `apache2`.

```
$ watch 'ps aux | grep apache2'
```

hoặc

```
$ watch "ps aux | grep apache2"
```

- Bây giờ, hãy mở trình duyệt web hoặc sử dụng trình duyệt dòng lệnh (chẳng hạn như `lynx`) để thiết lập kết nối với máy chủ web thông qua địa chỉ IP của nó. Bạn sẽ thấy gì trong đầu ra của `watch`?

  Một trong các tiến trình con do `www-data` sở hữu sẽ biến mất.

3. Như chúng ta đã biết, theo mặc định, `top` sẽ sắp xếp các tác vụ dựa vào phần trăm mức sử dụng CPU theo thứ tự giảm dần (giá trị cao hơn ở trên cùng). Hành vi này có thể được sửa đổi bằng các phím tương tác `M` (mức sử dụng bộ nhớ), `N` (mã định danh duy nhất của tiến trình), `T` (thời gian chạy) và `P` (phần trăm thời gian của CPU). Tuy nhiên, bạn cũng có thể sắp xếp danh sách tác vụ theo ý thích của mình bằng cách khởi chạy `top` với khóa chuyển `-o` (để biết thêm thông tin, hãy xem trang hướng dẫn của top). Bây giờ, hãy thực hiện các tác vụ sau:
   - Khởi chạy `top` để các tác vụ được sắp xếp theo mức sử dụng bộ nhớ:

```
$ top -o %MEM
```

   - Xác minh rằng bạn đã gõ đúng lệnh bằng cách bôi đen cột bộ nhớ:

     Nhấn `x`.

4. `ps` cũng có một khoá chuyển `o` để chỉ định các cột bạn muốn hiển thị. Hãy kiểm tra tùy chọn này và thực hiện các tác vụ sau:
   - Khởi chạy `ps` để chỉ hiển thị thông tin về *người dùng*, *phần trăm bộ nhớ đã sử dụng*, *phần trăm thời gian CPU đã sử dụng* và *lệnh đầy đủ*:

```
$ ps o user,%mem,%cpu,cmd
```

- Bây giờ, hãy khởi chạy `ps` để thông tin duy nhất được hiển thị là thông tin về người dùng và tên của chương trình họ đang sử dụng:

```
$ ps o user,comm
```

Linux Professional Institute

103.5 Bài 2

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.1 Tạo, giám sát và chấm dứt các Tiến trình
Bài:	2 trên 2

Giới thiệu

Các công cụ và tiện ích chúng ta đã thấy trong bài học trước rất hữu ích trong việc giám sát các tiến trình nói chung. Tuy nhiên, một quản trị viên hệ thống có thể sẽ cần nhiều hơn thế. Trong bài học này, chúng ta sẽ thảo luận về khái niệm bộ ghép kênh cửa sổ dòng lệnh cũng như về *Màn hình GNU* và *tmux* vì — dù các bộ giả lập cửa sổ dòng lệnh ngày nay đều rất hiện đại và hữu dụng — bộ ghép kênh vẫn bảo tồn một số tính năng mạnh mẽ và thú vị dành cho những quản trị viên hệ thống năng động.

Các tính năng của Bộ ghép kênh Cửa sổ Dòng lệnh

Trong các thiết bị điện tử, bộ ghép kênh (hoặc *mux*) là một thiết bị cho phép ta kết nối nhiều đầu vào với một đầu ra. Từ đó, bộ ghép kênh cửa sổ dòng lệnh sẽ cho phép chúng ta chuyển đổi giữa các đầu vào khác nhau theo yêu cầu. Mặc dù không hoàn toàn giống nhau, `screen` và `tmux` lại có cùng một loạt các tính năng chung:

- Bất kỳ một lần gọi hàm thành công nào cũng sẽ dẫn đến ít nhất một phiên bao gồm ít nhất một cửa sổ. Các cửa sổ này sẽ chứa các chương trình.

- Các cửa sổ có thể được chia thành các vùng hoặc ngăn — điều này có thể hỗ trợ năng suất khi làm việc đồng thời với nhiều chương trình khác nhau.
- Kiểm soát dễ dàng: để chạy hầu hết các lệnh, chúng sử dụng tổ hợp phím — cái được gọi là *tiền tố lệnh* hoặc *phím lệnh* — theo sau là một ký tự khác.
- Các phiên có thể được tách ra khỏi cửa sổ dòng lệnh của chúng (nghĩa là các chương trình sẽ được chuyển thành chạy ngầm và tiếp tục chạy). Điều này đảm bảo việc thực thi hoàn toàn các chương trình cho dù chúng ta vô tình đóng cửa sổ dòng lệnh, cửa sổ dòng lệnh đột nhiên bị đóng băng hoặc thậm chí là mất kết nối từ xa.
- Kết nối với ổ.
- Chế độ sao chép.
- Chúng có khả năng tùy biến cao.

Màn hình GNU

Trong những ngày đầu của Unix (những năm 1970 - 80), máy tính về cơ bản là một đơn vị chứa các cửa sổ dòng lệnh được kết nối với một máy tính trung tâm chứ không có nhiều cửa sổ hoặc tab. Và đó là lý do đằng sau việc tạo ra Màn hình GNU vào năm 1987: nó mô phỏng nhiều *màn hình VT100* độc lập trên một cửa sổ dòng lệnh vật lý duy nhất.

Cửa sổ

Màn hình GNU được gọi bằng cách gõ `screen` vào cửa sổ dòng lệnh. Trước tiên, chúng ta sẽ thấy một thông báo chào mừng:

```
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
(...)
```

Nhấn phím Dấu cách hoặc Enter để đóng thông báo và ta sẽ thấy dấu nhắc lệnh:

```
$
```

Có vẻ như không có gì xảy ra nhưng thực tế là `screen` đã tạo và quản lý phiên và cửa sổ đầu tiên

của nó. Tiền tố lệnh của màn hình là `Ctrl` + `a`. Để xem tất cả các cửa sổ ở cuối màn hình cửa sổ dòng lệnh, hãy nhập `Ctrl` + `a`-`w`:

```
0*$ bash
```

Đây chính là cửa sổ duy nhất của chúng ta cho đến thời điểm này! Tuy nhiên, hãy lưu ý rằng số đếm sẽ bắt đầu từ 0. Để tạo một cửa sổ khác, hãy nhập `Ctrl` + `a`-`c` và ta sẽ thấy một dấu nhắc lệnh mới. Hãy cùng bắt đầu `ps` trong cửa sổ mới đó:

```
$ ps
  PID TTY          TIME CMD
  974 pts/2    00:00:00 bash
  981 pts/2    00:00:00 ps
```

và gõ lại `Ctrl` + `a`-`w`:

```
0-$ bash  1*$ bash
```

Ở đây chúng ta có hai cửa sổ (lưu ý dấu hoa thị cho biết cửa sổ đang được hiển thị vào lúc này). Tuy nhiên, khi bắt đầu với Bash, cả hai đều sẽ được đặt tên giống nhau. Vì đã gọi `ps` trong cửa sổ hiện tại nên hãy đổi tên nó thành cùng một tên đó. Để làm được điều này, ta phải nhập `Ctrl` + `a`-`A` và tên cửa sổ mới (`ps`) khi được nhắc:

```
Set window's title to: ps
```

Bây giờ, hãy tạo một cửa sổ khác nhưng đặt tên cho nó ngay từ đầu là `yetanotherwindow`. Điều này có thể được thực hiện bằng cách gọi `screen` với khoá chuyển `-t`:

```
$ screen -t yetanotherwindow
```

Ta có thể di chuyển giữa các cửa sổ bằng nhiều cách khác nhau:

- Bằng cách sử dụng `Ctrl` + `a`-`n` (đi tới cửa sổ *tiếp theo*) và `Ctrl` + `a`-`p` (đi tới cửa sổ *trước đó*).
- Bằng cách sử dụng `Ctrl` + `a`-`number` (đi tới cửa sổ số *number*).
- Bằng cách sử dụng `Ctrl` + `a`-`"` để xem danh sách tất cả các cửa sổ. Ta có thể di chuyển lên và xuống bằng các phím mũi tên và chọn cửa sổ mình muốn bằng cách nhấn Enter:

```
Num Name                         Flags

   0 bash                                                    $
   1 ps                                                      $
   2 yetanotherwindow
```

Khi làm việc với các cửa sổ, điều quan trọng cần nhớ là:

- Cửa sổ sẽ chạy các chương trình của chúng một cách hoàn toàn độc lập.
- Các chương trình sẽ tiếp tục chạy ngay cả khi cửa sổ của chúng không được hiển thị (và cả khi phiên màn hình được tách ra như chúng ta sẽ thấy ngay sau đây).

Để xóa một cửa sổ, ta chỉ cần chấm dứt chương trình đang chạy trong đó (khi cửa sổ cuối cùng bị xóa, `screen` sẽ tự chấm dứt). Ngoài ra, hãy sử dụng `Ctrl` + `a`-`k` trong cửa sổ cần muốn xóa; chúng ta sẽ được yêu cầu xác nhận:

```
Really kill this window [y/n]

Window 0 (bash) killed.
```

Vùng

`screen` có thể chia màn hình cửa sổ dòng lệnh thành nhiều vùng để chứa các cửa sổ. Các phần chia này có thể nằm ngang (`Ctrl` + `a`-`S`) hoặc dọc (`Ctrl` + `a`-`|`).

Điều duy nhất mà vùng mới hiển thị sẽ chỉ là `--` ở dưới cùng mang ý nghĩa là nó đang trống:

```
   1 ps                                                      --
```

Để chuyển sang vùng mới, hãy nhập `Ctrl` + `a`-`Tab`. Bây giờ, ta có thể thêm một cửa sổ bằng bất kỳ phương pháp nào mà chúng ta đã biết (ví dụ như `Ctrl` + `a`-`2`). Bây giờ, `--` sẽ biến thành `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                                                    2 yetanotherwindow
```

Các khía cạnh quan trọng cần ghi nhớ khi làm việc với các vùng là:

- Di chuyển giữa các vùng bằng cách gõ `Ctrl` + `a` - `Tab`.
- Có thể chấm dứt tất cả các vùng ngoại trừ vùng hiện tại bằng `Ctrl` + `a` - `Q`.
- Có thể chấm dứt vùng hiện tại bằng `Ctrl` + `a` - `X`.
- Chấm dứt một vùng sẽ không chấm dứt cửa sổ của nó.

Phiên

Cho đến nay, chúng ta đã làm quen với một vài cửa sổ và vùng, nhưng tất cả trong số chúng đều thuộc về cùng một phiên duy nhất. Đã đến lúc bắt đầu làm quen với các phiên. Để xem danh sách tất cả các phiên, hãy nhập `screen -list` hoặc `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.
```

Đó là phiên duy nhất của chúng ta cho đến nay:

PID
: `1037`

Tên
: `pts-0.debian` (biểu thị cửa sổ dòng lệnh — trong trường hợp của chúng ta là *pseudo terminal slave* — và tên máy chủ).

Trạng thái
: Đính kèm (`Attached`).

Hãy cùng tạo một phiên mới và đặt cho nó một cái tên mô tả rõ ràng hơn:

```
$ screen -S "second session"
```

Màn hình cửa sổ dòng lệnh sẽ bị xóa và ta sẽ nhận được một dấu nhắc mới. Chúng taa có thể kiểm tra các phiên một lần nữa:

```
$ 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.
```

Để kết thúc một phiên, hãy thoát khỏi tất cả các cửa sổ của nó hoặc chỉ cần gõ lệnh `screen -S SESSION-PID -X quit` (ta cũng có thể cung cấp tên phiên thay thế). Hãy cùng thoát khỏi phiên đầu tiên:

```
$ screen -S 1037 -X quit
```

Ta sẽ được đưa trở lại dấu nhắc trong cửa sổ dòng lệnh bên ngoài `screen`. Nhưng hãy nhớ rằng, phiên thứ hai của chúng ta vẫn còn tồn tại:

```
$ screen -ls
There is a screen on:
    1090.second session (08/24/19 14:38:35) (Detached)
1 Socket in /run/screen/S-carol.
```

Tuy nhiên, vì đã hủy phiên mẹ nên nó đã được cấp một nhãn mới là `Detached` (Rời).

Tách Phiên

Vì một số lý do, ta có thể sẽ muốn tách phiên màn hình khỏi cửa sổ dòng lệnh của nó:

- Để máy tính tại nơi làm việc thực hiện công việc của nó và sau này kết nối từ xa tại nhà.
- Chia sẻ một phiên với những người dùng khác.

Ta có thể tách một phiên bằng tổ hợp phím `Ctrl` + `a` - `d`. Ta sẽ được đưa trở lại cửa sổ dòng lệnh của mình:

```
[detached from 1090.second session]
$
```

Để đính lại phiên, ta sẽ sử dụng lệnh `screen -r SESSION-PID`. Ngoài ra, ta có thể sử dụng

`SESSION-NAME` như đã thấy ở trên. Nếu chỉ có một phiên tách rời, cả hai đều không phải là bắt buộc:

```
$ screen -r
```

Lệnh này đủ để đính vào lại phiên thứ hai:

```
$ screen -ls
There is a screen on:
        1090.second session      (08/24/19 14:38:35)     (Attached)
1 Socket in /run/screen/S-carol.
```

Các tùy chọn quan trọng để đính lại phiên:

`-d -r`

Đính lại một phiên và — nếu cần — tách nó ra trước.

`-d -R`

Tương tự như `-d -r` nhưng `screen` thậm chí sẽ tạo phiên đầu tiên nếu nó không tồn tại.

`-d -RR`

Tương tự như `-d -R`. Tuy nhiên, tuỳ chọn này sẽ sử dụng phiên đầu tiên nếu có nhiều phiên.

`-D -r`

Đính lại một phiên. Nếu cần, hãy tách và đăng xuất từ xa trước.

`-D -R`

Nếu một phiên đang chạy, hãy đính lại (tách và đăng xuất từ xa trước nếu cần). Nếu nó không chạy, hãy tạo nó và thông báo cho người dùng.

`-D -RR`

Tương tự như `-D -R` — nhưng mạnh hơn.

`-d -m`

Bắt đầu `screen` ở chế độ *tách*. Tuỳ chọn này sẽ tạo ra một phiên mới nhưng không đính kèm với nó. Tuỳ chọn này hữu ích đối với các tệp lệnh khởi động hệ thống.

`-D -m`

Tương tự như `-d -m` nhưng không rẽ nhánh một tiến trình mới. Lệnh sẽ thoát nếu phiên kết thúc.

Hãy đọc trang hướng dẫn dành cho `screen` để tìm hiểu về các tùy chọn khác.

Sao chép & Dán: Chế độ cuộn lại

Màn hình GNU có chế độ sao chép hoặc *cuộn lại*. Sau khi vào, ta có thể di chuyển con trỏ trong cửa sổ hiện tại và đi qua nội dung lịch sử của nó bằng các phím mũi tên. Ta có thể đánh dấu văn bản và sao chép nó qua các cửa sổ. Các bước để làm theo là:

1. Vào chế độ sao chép/cuộn lại: `Ctrl` + `a`-`[`.
2. Di chuyển đến đầu đoạn văn bản cần sao chép bằng các phím mũi tên.
3. Đánh dấu phần đầu của đoạn văn bản cần sao chép: Dấu cách.
4. Di chuyển đến cuối đoạn văn bản cần sao chép bằng các phím mũi tên.
5. Đánh dấu phần cuối của đoạn văn bản cần sao chép: Dấu cách.
6. Chuyển đến cửa sổ muốn chọn và dán đoạn văn bản: `Ctrl` + `a`-`]`.

Tùy chỉnh Màn hình

Tệp cấu hình toàn hệ thống cho màn hình là `/etc/screenrc`. Ngoài ra, ta có thể sử dụng một lệnh `~/.screenrc` cấp người dùng. Tệp bao gồm bốn phần cấu hình chính:

`SCREEN SETTINGS`

Ta có thể xác định cài đặt chung bằng cách chỉ định *chỉ thị*, theo sau là khoảng trắng và *giá trị* (như trong `defscrollback 1024`).

`SCREEN KEYBINDINGS`

Phần này khá thú vị vì nó cho phép ta xác định lại các tổ hợp phím có thể can thiệp vào việc sử dụng cửa sổ dòng lệnh thường xuyên. Sử dụng từ khóa `bind`, theo sau là một khoảng trắng, ký tự sẽ sử dụng sau tiền tố lệnh, một khoảng trắng khác và lệnh sẽ là `bind l kill` (cài đặt này sẽ thay đổi cách tắt cửa sổ mặc định thành `Ctrl` + `a`-`l`).

Để hiển thị tất cả các liên kết của màn hình, hãy nhập `Ctrl` + `a`-`?` hoặc tham khảo trang hướng dẫn.

> **TIP** Tất nhiên, bạn cũng có thể thay đổi tiền tố lệnh. Ví dụ: để chuyển từ `Ctrl` + `a` sang `Ctrl` + `b`, bạn chỉ cần thêm dòng này: `escape ^Bb`.

`TERMINAL SETTINGS`

Phần này bao gồm các cài đặt liên quan đến kích thước cửa sổ của cửa sổ dòng lệnh và bộ nhớ đệm bên cạnh các cài đặt khác. Ví dụ: để bật chế độ không chặn nhằm đối phó tốt hơn với các

kết nối ssh không ổn định, cấu hình được sử dụng sẽ là `defnonblock 5`.

`STARTUP SCREENS`

Ta có thể bao gồm cả các lệnh để chạy các chương trình khác nhau khi khởi động `screen`, ví dụ như `screen -t top top` (màn hình sẽ mở một cửa sổ có tên `top` với `top` ở bên trong).

tmux

`tmux` được phát hành vào năm 2007. Mặc dù rất giống với `screen` nhưng nó lại có một vài điểm khác biệt đáng chú ý:

- Mô hình máy khách-máy chủ: máy chủ cung cấp một số phiên, mỗi phiên có thể có một số cửa sổ được liên kết với nó và từ đó có thể được chia sẻ bởi nhiều máy khách khác nhau.
- Chọn lọc tương tác các phiên, cửa sổ và ứng dụng khách thông qua các menu.
- Cùng một cửa sổ có thể được liên kết với nhiều phiên.
- Có sẵn cả hai bố cục phím *vim* và *Emacs*.
- Hỗ trợ thiết bị cửa sổ dòng lệnh UTF-8 và 256 màu.

Cửa sổ

`tmux` có thể được gọi chỉ bằng cách gõ `tmux` tại dấu nhắc lệnh. Ta sẽ thấy một dấu nhắc vỏ và một thanh trạng thái ở cuối cửa sổ:

```
[0] 0:bash*                                                    "debian" 18:53 27-Aug-19
```

Ngoài `hostname` (tên máy chủ), thời gian và ngày tháng, thanh trạng thái còn cung cấp các thông tin sau:

Tên Phiên

`[0]`

Số lượng Cửa sổ

`0:`

Tên cửa sổ

`bash*`. Theo mặc định, đây là tên của chương trình đang chạy bên trong cửa sổ và — không giống như `screen` — `tmux` sẽ tự động cập nhật nó để thể hiện chương trình đang chạy hiện tại. Hãy lưu ý dấu hoa thị cho ta biết đây là cửa sổ hiện tại và có thể nhìn thấy.

Ta có thể gán tên phiên và tên cửa sổ khi gọi `tmux`:

```
$ tmux new -s "LPI" -n "Window zero"
```

Thanh trạng thái sẽ thay đổi tương ứng:

```
[LPI] 0:Window zero*                                          "debian" 19:01 27-Aug-
19
```

Tiền tố lệnh của tmux là `Ctrl` + `b`. Để tạo một cửa sổ mới, ta chỉ cần gõ `Ctrl` + `b`-`c`. Ta sẽ được đưa đến một dấu nhắc mới và thanh trạng thái sẽ thể hiện cửa sổ mới:

```
[LPI] 0:Window zero- 1:bash*                                  "debian" 19:02 27-Aug-
19
```

Vì Bash là vỏ nền tảng nên cửa sổ mới sẽ được đặt tên đó theo mặc định. Hãy bắt đầu lệnh `top` và xem tên cửa sổ được thay đổi thành `top`:

```
[LPI] 0:Window zero- 1:top*                                   "debian" 19:03 27-Aug-19
```

Trong mọi trường hợp, ta đều có thể đổi tên cửa sổ bằng `Ctrl` + `b`-`,`. Khi được nhắc, hãy cung cấp tên mới và nhấn Enter:

```
(rename-window) Window one
```

Ta có thể hiển thị tất cả các cửa sổ để lựa chọn với `Ctrl` + `b`-`w` (sử dụng các phím mũi tên để di chuyển lên xuống và `enter` để chọn):

```
(0)  0: Window zero- "debian"
(1)  1: Window one* "debian"
```

Theo cách tương tự với `screen`, chúng ta có thể chuyển từ cửa sổ này sang cửa sổ khác bằng:

`Ctrl` + **`b`**-**`n`**
: đi tới cửa sổ *tiếp theo*.

`Ctrl` + `b` - `p`
: đi tới cửa sổ *trước đó*.

`Ctrl` + `b` - `number`
: đi tới cửa sổ số *number*.

Để thoát khỏi cửa sổ, hãy sử dụng `Ctrl` + `b` - `&`. Ta sẽ được yêu cầu xác nhận:

```
kill-window Window one? (y/n)
```

Các lệnh cửa sổ thú vị khác gồm có:

`Ctrl` + `b` - `f`
: tìm cửa sổ theo tên.

`Ctrl` + `b` - `.`
: thay đổi số chỉ mục cửa sổ.

Để đọc toàn bộ danh sách các lệnh, hãy tham khảo trang hướng dẫn.

Ngăn

Tiện ích chia cửa sổ của `screen` cũng có trong `tmux`. Tuy nhiên, kết quả của việc phân chia lại không được gọi là *vùng* mà là *ngăn*. Khác với vùng, ngăn là các cửa sổ dòng lệnh giả hoàn chỉnh được liên kết với một cửa sổ. Điều này có nghĩa là việc tắt một ngăn cũng sẽ tắt cả cửa sổ dòng lệnh giả của nó và bất kỳ chương trình liên quan nào đang chạy bên trong.

Để chia một cửa sổ theo chiều ngang, chúng ta sử dụng `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
KiB Mem :  4050960 total,  3730920 free,   114880 used,   205160 buff/cache
KiB Swap:  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*                                  "debian" 19:05 27-
Aug-19
```

Để chia theo chiều dọc, hãy sử dụng `Ctrl` + `b` - `%`:

```
                                                                          |$
    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-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
```

Để hủy ngăn hiện tại (cùng với cửa sổ dòng lệnh giả đang chạy bên trong ngăn đó và bất kỳ chương trình liên quan nào), hãy sử dụng `Ctrl` + `b` - `x`. Ta sẽ được yêu cầu xác nhận trên thanh trạng thái:

```
kill-pane 1? (y/n)
```

Các lệnh Ngăn quan trọng:

`Ctrl` + **`b`** - **`↑`**, **`↓`**, **`←`**, **`→`**
: di chuyển giữa các ngăn.

`Ctrl` + **`b`** - **`;`**
: di chuyển đến ngăn hoạt động cuối cùng.

`Ctrl` + **`b`** - **`Ctrl`** + **`arrow key`**
: thay đổi kích thước ngăn bằng một dòng.

`Ctrl` + `b` - `Altl` + `arrow key`
thay đổi kích thước ngăn bằng năm dòng.

`Ctrl` + `b` - `{`
hoán đổi ngăn (ngăn hiện tại sang ngăn trước đó).

`Ctrl` + `b` - `}`
hoán đổi ngăn (ngăn hiện tại sang ngăn tiếp theo).

`Ctrl` + `b` - `z`
phóng to/thu nhỏ ngăn.

`Ctrl` + `b` - `t`
`tmux` hiển thị một chiếc đồng hồ bên trong ngăn (tắt nó bằng cách nhấn `q`).

`Ctrl` + `b` - `!`
biến ngăn thành cửa sổ.

Để đọc toàn bộ danh sách các lệnh, hãy tham khảo trang hướng dẫn.

Phiên

Để liệt kê các phiên trong `tmux`, ta có thể sử dụng `Ctrl` + `b` - `s`:

```
(0) + LPI: 2 windows (attached)
```

Ngoài ra, ta có thể sử dụng lệnh `tmux ls`:

```
$ tmux ls
LPI: 2 windows (created Tue Aug 27 19:01:49 2019) [158x39] (attached)
```

Chỉ có một phiên (`LPI`) là có bao gồm hai cửa sổ. Hãy cùng tạo một phiên mới từ trong phiên hiện tại của chúng ta. Điều này có thể đạt được bằng cách sử dụng `Ctrl` + `b`, nhập `:new` tại dấu nhắc, sau đó nhấn Enter. Ta sẽ được chuyển đến phiên làm việc mới như có thể quan sát thấy được trên thanh trạng thái:

```
[2] 0:bash*                                                    "debian" 19:15 27-Aug-19
```

Theo mặc định, `tmux` sẽ đặt tên phiên là `2`. Để đổi tên nó, hãy sử dụng `Ctrl` + `b` - `$`. Khi được nhắc, hãy

cung cấp tên mới và nhấn Enter:

```
(rename-session) Second Session
```

Ta có thể chuyển phiên bằng `Ctrl` + `b` - `s` (sử dụng các phím mũi tên và `enter`):

```
(0) + LPI: 2 windows
(1) + Second Session: 1 windows (attached)
```

Để hủy phiên, ta có thể sử dụng lệnh `tmux kill-session -t SESSION-NAME`. Nếu nhập lệnh từ bên trong phiên đính kèm hiện tại, ta sẽ được đưa ra khỏi `tmux` và quay lại phiên cửa sổ dòng lệnh ban đầu:

```
$ tmux kill-session -t "Second Session"
[exited]
$
```

Tách Phiên

Bằng cách chấm dứt `Second Session`, chúng ta đã bị thoát ra ngoài `tmux`. Tuy nhiên, chúng ta vẫn có một phiên đang hoạt động. Hãy hỏi `tmux` để biết danh sách các phiên và chắc chắn chúng ta sẽ tìm thấy nó ở đó:

```
$ tmux ls
LPI: 2 windows (created Tue Aug 27 19:01:49 2019) [158x39]
```

Tuy nhiên, phiên này đã được tách ra khỏi cửa sổ dòng lệnh của nó. Chúng ta có thể đính nó vào bằng `tmux attachment -t SESSION-NAME` (`attach` có thể được thay thế bằng `at` hoặc — đơn giản — `a`). Khi chỉ có một phiên duy nhất, đặc tả tên là tùy chọn:

```
$ tmux a
```

Bây giờ, ta đã trở lại phiên làm việc của mình; để tách khỏi nó, hãy nhấn `Ctrl` + `b` - `d`:

```
[detached (from session LPI)]
$
```

TIP

Cùng một phiên có thể được đính vào nhiều cửa sổ dòng lệnh. Nếu muốn đính vào một phiên để đảm bảo rằng phiên đầu tiên sẽ được tách ra khỏi bất kỳ một cửa sổ dòng lệnh nào khác, hãy sử dụng khóa chuyển `-d`: `tmux attachment -d -t SESSION-NAME`.

Các lệnh quan trọng để đính/tách phiên:

`Ctrl` + `b` - `D`
: chọn máy khách cần tách.

`Ctrl` + `b` - `r`
: làm mới cửa sổ dòng lệnh của máy khách.

Để đọc toàn bộ danh sách các lệnh, hãy tham khảo trang hướng dẫn.

Sao chép & Dán: Chế độ cuộn lại

`tmux` cũng có chế độ sao chép về cơ bản là giống như `screen` (hãy nhớ sử dụng tiền tố lệnh của tmux chứ không phải của `screen`). Sự khác biệt duy nhất về mặt lệnh là ta sẽ sử dụng `Ctrl` + Dấu cách để đánh dấu phần đầu của vùng chọn và `Alt` + `w` để sao chép văn bản đã chọn.

Tùy chỉnh tmux

Các tệp cấu hình cho `tmux` thường nằm ở `/etc/tmux.conf` và `~/.tmux.conf`. Khi bắt đầu, `tmux` sẽ cung cấp các tệp này nếu chúng tồn tại. Ngoài ra còn có một khả năng khác là bắt đầu `tmux` với khóa chuyển `-f` để cung cấp tệp cấu hình thay thế. Ta có thể tìm thấy ví dụ về tệp cấu hình `tmux` tại `/usr/share/doc/tmux/example_tmux.conf`. Mức độ tùy chỉnh mà ta có thể đạt được thực sự rất cao. Một số điều chúng ta có thể làm bao gồm:

- Thay đổi khóa tiền tố

```
# Change the prefix key to C-a
set -g prefix C-a
unbind C-b
bind C-a send-prefix
```

- Đặt thêm tổ hợp phím cho cửa sổ cao hơn 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
```

Để có danh sách đầy đủ tất cả các liên kết, hãy nhập `Ctrl` + `b` - `?` (nhấn `q` để thoát) hoặc tham khảo trang hướng dẫn.

Bài tập Hướng dẫn

Hãy cho biết các câu lệnh/tính năng nào sau đây tương ứng với GNU Screen, tmux hoặc cả hai:

+

Tính năng/Câu lệnh	Màn hình GNU	tmux
Tiền tố lệnh mặc định là `Ctrl` + `a`		
Mô hình Máy Khách-Máy Chủ		
Ngăn là cửa sổ dòng lệnh giả		
Chấm dứt một vùng mà không chấm dứt các cửa sổ liên quan		
Phiên bao gồm các cửa sổ		
Phiên có thể được tách ra		

1. Hãy cài đặt GNU Screen trên máy tính của bạn (tên gói: `screen`) và hoàn thành các tác vụ sau:
   - Khởi động chương trình. Bạn sẽ sử dụng lệnh gì?
   - Bắt đầu `top`.
   - Sử dụng tiền tố khóa của screen và mở một cửa sổ mới; sau đó, mở `/etc/screenrc` bằng `vi`.
   - Liệt kê các cửa sổ ở cuối màn hình.
   - Đổi tên của cửa sổ hiện tại thành `vi`.
   - Đổi tên cửa sổ còn lại thành `top`. Để làm điều đó, trước tiên hãy hiển thị danh sách tất cả các cửa sổ để bạn có thể di chuyển lên xuống và chọn cửa sổ phù hợp.
   - Kiểm tra xem tên cửa sổ đã được thay đổi chưa bằng cách hiển thị lại tên cửa sổ ở cuối màn

hình.

- Bây giờ, hãy tách phiên ra và để `screen` tạo một phiên mới có tên `ssh`.

- Cũng tách khỏi `ssh` và để `screen` hiển thị danh sách các phiên.

- Bây giờ, hãy đính vào phiên đầu tiên bằng cách sử dụng PID của nó.

- Bạn nên quay lại cửa sổ hiển thị `top`. Chia cửa sổ theo chiều ngang và di chuyển đến vùng trống mới.

- Để `screen` liệt kê tất cả các cửa sổ và chọn `vi` để hiển thị trong vùng trống mới.

- Bây giờ, hãy chia vùng hiện tại theo chiều dọc, di chuyển vào vùng trống mới được tạo và liên kết nó với một cửa sổ hoàn toàn mới.

- Chấm dứt tất cả các vùng ngoại trừ vùng hiện tại (hãy nhớ rằng, mặc dù đã chấm dứt các vùng nhưng các cửa sổ vẫn còn hoạt động). Sau đó, thoát khỏi tất cả các cửa sổ của phiên hiện tại cho đến khi phiên đó kết thúc.

- Cuối cùng, yêu cầu `screen` liệt kê các phiên của nó một lần nữa, chấm dứt phiên `ssh` còn lại bằng PID và kiểm tra xem có thực sự là không còn phiên nào nữa hay không.

2. Hãy cài đặt `tmux` trên máy tính của bạn (tên gói: `tmux`) và hoàn thành các tác vụ sau:
   - Khởi động chương trình. Bạn sẽ sử dụng lệnh gì?

- Bắt đầu `top` (hãy lưu ý — trong vài giây — tên của cửa sổ thay đổi thành `top` trong thanh trạng thái):

- Sử dụng tiền tố chính của tmux, mở một cửa sổ mới; sau đó, tạo `~/.tmux.conf` bằng cách sử dụng `nano`:

- Chia đôi cửa sổ theo chiều dọc và giảm kích thước của ngăn vừa tạo một vài lần:

- Đổi tên của cửa sổ hiện tại thành `text editing`; sau đó, yêu cầu `tmux` hiển thị danh sách tất cả các phiên của nó:

- Di chuyển đến cửa sổ đang chạy `top` và quay lại cửa sổ hiện tại bằng cùng một tổ hợp phím:

- Tách khỏi phiên hiện tại và tạo một phiên mới có tên là `ssh` và tên cửa sổ của nó là `ssh window`:

- Cũng tách khỏi phiên `ssh` và để `tmux` hiển thị lại danh sách các phiên:

NOTE

========== ========== Từ thời điểm này trở đi, bài tập sẽ yêu cầu bạn sử dụng máy *từ xa* để kết nối `ssh` với máy chủ lưu trữ cục bộ của bạn (máy ảo hoàn toàn hợp lệ và có thể sẽ rất hữu ích). Hãy đảm bảo rằng bạn đã cài đặt và chạy `openssh-server` trên máy cục bộ của mình và ít nhất là `openssh-client` đã được cài đặt trên máy từ xa. ========== ==========

- Bây giờ, hãy khởi động một máy từ xa và kết nối qua `ssh` với máy chủ cục bộ của bạn. Khi kết nối đã được thiết lập, hãy kiểm tra các phiên `tmux`:

- Trên máy chủ từ xa, hãy đính phiên `ssh` theo tên:

- Quay lại máy cục bộ của bạn, hãy đính vào phiên `ssh` theo tên để đảm bảo kết nối với máy chủ từ xa sẽ bị ngắt trước:

- Hiển thị tất cả các phiên để lựa chọn và chuyển đến phiên đầu tiên của bạn (`[0]`). Sau đó, hãy chấm dứt phiên `ssh` theo tên:

- Cuối cùng, tách khỏi phiên hiện tại và chấm dứt nó theo tên:

Bài tập Mở rộng

1. Cả `screen` và `tmux` đều có thể vào chế độ dòng lệnh thông qua *tiền tố lệnh* + `:` (chúng ta đã thấy một ví dụ ngắn gọn với `tmux`). Hãy tìm hiểu và thực hiện các tác vụ sau trong chế độ dòng lệnh:
   - Đưa `screen` vào chế độ sao chép:
   - Buộc `tmux` đổi tên cửa sổ hiện tại:
   - Buộc `screen` đóng tất cả các cửa sổ và kết thúc phiên:
   - Buộc `tmux` chia một khung thành hai:
   - Buộc `tmux` chấm dứt cửa sổ hiện tại:
2. Khi vào chế độ sao chép trong `screen`, không chỉ có các phím mũi tên và phím `PgUP` hoặc `PgDown` để điều hướng cửa sổ hiện tại và bộ đệm cuộn ngược mà còn có trình chỉnh sửa toàn màn hình giống vi. Bằng cách sử dụng trình chỉnh sửa này, hãy thực hiện các tác vụ sau:
   - Echo `supercalifragilisticexpialidocious` trong cửa sổ dòng lệnh `screen`:
   - Sao chép năm ký tự liên tiếp (từ trái sang phải) ở dòng ngay phía trên của con trỏ:
   - Cuối cùng, dán lại vùng đã chọn (`stice`) vào dấu nhắc lệnh:
3. Giả sử bạn muốn chia sẻ phiên `tmux` (`our_session`) với người dùng khác. Bạn đã tạo ổ nối (`/tmp/our_socket`) với các quyền phù hợp để cả bạn và người dùng khác đều có thể đọc và ghi được. Cần đáp ứng hai điều kiện nào khác để người dùng thứ hai có thể đính thành công phiên thông qua `tmux -S /tmp/our_socket a -t our_session`?

Tóm tắt

Trong bài học này, chúng ta đã học về *bộ ghép kênh cửa sổ dòng lệnh* nói chung và GNU Screen và tmux nói riêng. Các khái niệm quan trọng cần nhớ bao gồm:

- Tiền tố lệnh: `screen` sử dụng `Ctrl` + `a` + *ký tự*; `tmux`, `Ctrl` + `b` + _ký tự.
- Cấu trúc của phiên, cửa sổ và phân chia cửa sổ (vùng hoặc ngăn).
- Chế độ sao chép.
- Tách phiên: một trong những tính năng mạnh nhất của bộ ghép kênh.

Các lệnh được dùng trong bài học này:

`screen`
: Bắt đầu một phiên `screen`.

`tmux`
: Bắt đầu một phiên `tmux`.

Đáp án Bài tập Hướng dẫn

Hãy cho biết các câu lệnh/tính năng nào sau đây tương ứng với GNU Screen, tmux hoặc cả hai:

+

Tính năng/Câu lệnh	Màn hình GNU	tmux
Tiền tố lệnh mặc định là `Ctrl` + `a`	x	
Mô hình Máy Khách-Máy Chủ		x
Ngăn là cửa sổ dòng lệnh giả		x
Chấm dứt một vùng mà không chấm dứt (các) cửa sổ liên quan	x	
Phiên bao gồm các cửa sổ	x	x
Phiên có thể được tách ra	x	x

1. Hãy cài đặt GNU Screen trên máy tính của bạn (tên gói: `screen`) và hoàn thành các tác vụ sau:
   - Khởi động chương trình. Bạn sẽ sử dụng lệnh gì?

     `screen`
   - Bắt đầu `top`.

     `top`
   - Sử dụng tiền tố khóa của screen và mở một cửa sổ mới; sau đó, mở `/etc/screenrc` bằng `vi`.

     `Ctrl` + `a`-`c`

     `sudo vi /etc/screenrc`
   - Liệt kê các cửa sổ ở cuối màn hình.

     `Ctrl` + `a`-`w`
   - Đổi tên của cửa sổ hiện tại thành `vi`.

     `Ctrl` + `a`-`A`. Sau đó, chúng ta phải gõ `vi` và nhấn `enter`.
   - Đổi tên cửa sổ còn lại thành `top`. Để làm điều đó, trước tiên hãy hiển thị danh sách tất cả các cửa sổ để bạn có thể di chuyển lên xuống và chọn cửa sổ phù hợp.

Trước tiên, hãy nhập `Ctrl` + `a`-`"`. Sau đó, sử dụng các phím mũi tên để đánh dấu cửa sổ có nội dung `0 bash` và nhấn `enter`. Cuối cùng, nhập `Ctrl` + `a`-`A`, nhập `top` và nhấn `enter`.

- Kiểm tra xem tên cửa sổ đã được thay đổi chưa bằng cách hiển thị lại tên cửa sổ ở cuối màn hình.

  `Ctrl` + `a`-`w`

- Bây giờ, hãy tách phiên ra và để `screen` tạo một phiên mới có tên `ssh`.

  `Ctrl` + `a`-`d` `screen -S "ssh"` và nhấn `enter`.

- Cũng tách khỏi `ssh` và để `screen` hiển thị danh sách các phiên.

  `Ctrl` + `a`-`d` `screen -list` hoặc `screen -ls`.

- Bây giờ, hãy đính vào phiên đầu tiên bằng cách sử dụng PID của nó.

  `screen -r PID-OF-SESSION`

- Bạn nên quay lại cửa sổ hiển thị `top`. Chia cửa sổ theo chiều ngang và di chuyển đến vùng trống mới.

  `Ctrl` + `a`-`S`

  `Ctrl` + `a`-`Tab`

- Để `screen` liệt kê tất cả các cửa sổ và chọn `vi` để hiển thị trong vùng trống mới.

  Sử dụng `Ctrl` + `a`-`"` để hiển thị tất cả các cửa sổ để lựa chọn, đánh dấu `vi` và nhấn `enter`.

- Bây giờ, hãy chia vùng hiện tại theo chiều dọc, di chuyển vào vùng trống mới được tạo và liên kết nó với một cửa sổ hoàn toàn mới.

  `Ctrl` + `a`-`|`

  `Ctrl` + `a`-`Tab`

  `Ctrl` + `a`-`c`

- Chấm dứt tất cả các vùng ngoại trừ vùng hiện tại (hãy nhớ rằng, mặc dù đã chấm dứt các vùng nhưng các cửa sổ vẫn còn hoạt động). Sau đó, thoát khỏi tất cả các cửa sổ của phiên hiện tại cho đến khi phiên đó kết thúc.

  `Ctrl` + `a`-`Q`. `exit` (để thoát khỏi Bash). `Shift` + `:`, sau đó, gõ `quit` và nhấn `enter` (để thoát `vi`). Sau

đó, hãy gõ `exit` (để thoát khỏi vỏ Bash bên dưới) `q` (để kết thúc `top`); sau đó, hãy gõ `exit` (để thoát khỏi vỏ Bash bên dưới).

- Cuối cùng, yêu cầu `screen` liệt kê các phiên của nó một lần nữa, chấm dứt phiên `ssh` còn lại bằng PID và kiểm tra xem có thực sự là không còn phiên nào nữa hay không.

  `screen -list` or `screen -ls`

  `screen -S PID-OF-SESSION -X quit`

  `screen -list` or `screen -ls`

2. Hãy cài đặt `tmux` trên máy tính của bạn (tên gói: `tmux`) và hoàn thành các tác vụ sau:
   - Khởi động chương trình. Bạn sẽ sử dụng lệnh gì?

     + `tmux`

   - Bắt đầu `top` (hãy lưu ý — trong vài giây — tên của cửa sổ thay đổi thành `top` trong thanh trạng thái):

     `top`

   - Sử dụng tiền tố chính của tmux, mở một cửa sổ mới; sau đó, tạo `~/.tmux.conf` bằng cách sử dụng `nano`:

     `Ctrl` + `b` - `c` `nano ~/.tmux.conf`

   - Chia đôi cửa sổ theo chiều dọc và giảm kích thước của ngăn vừa tạo một vài lần:

     `Ctrl` + `b` - `"`

     `Ctrl` + `b` - `Ctrl` + `↓`

   - Đổi tên của cửa sổ hiện tại thành `text editing`; sau đó, yêu cầu `tmux` hiển thị một danh sách với tất cả các phiên của nó:

     `Ctrl` + `b` - `,`. Sau đó, chúng ta sẽ cung cấp tên mới và nhấn `enter`. `Ctrl` + `b` - `s` hoặc `tmux ls`.

   - Di chuyển đến cửa sổ đang chạy `top` và quay lại cửa sổ hiện tại bằng cùng một tổ hợp phím:

     `Ctrl` + `b` - `n` hoặc `Ctrl` + `b` - `p`

   - Tách khỏi phiên hiện tại và tạo một phiên mới có tên là `ssh` với tên cửa sổ là `ssh window`:

`Ctrl` + `b` - `d` `tmux new -s "ssh" -n "ssh window"`

- Cũng tách khỏi phiên `ssh` và để `tmux` hiển thị lại danh sách các phiên:

  `Ctrl` + `b` - `d` `tmux ls`

  NOTE

  `==========` Từ thời điểm này trở đi, bài tập sẽ yêu cầu bạn sử dụng máy *từ xa* để kết nối `ssh` với máy chủ lưu trữ cục bộ của bạn (máy ảo hoàn toàn hợp lệ và có thể sẽ rất hữu ích). Hãy đảm bảo rằng bạn đã cài đặt và chạy `openssh-server` trên máy cục bộ của mình và ít nhất là `openssh-client` đã được cài đặt trên máy từ xa. `==========`

- Bây giờ, hãy khởi động một máy từ xa và kết nối qua `ssh` với máy chủ cục bộ của bạn. Khi kết nối đã được thiết lập, hãy kiểm tra các phiên `tmux`:

  Trên máy chủ từ xa: `ssh local-username@local-ipaddress`. Sau khi kết nối với máy cục bộ: `tmux ls`.

- Trên máy chủ từ xa, hãy đính phiên `ssh` theo tên:

  `tmux a -t ssh` (`a` có thể được thay thế bằng `at` hoặc `attach`).

- Quay lại máy cục bộ của bạn, hãy đính vào phiên `ssh` theo tên để đảm bảo kết nối với máy chủ từ xa sẽ bị ngắt trước:

  `tmux a -d -t ssh` (`a` có thể được thay thế bằng `at` hoặc `attach`).

- Hiển thị tất cả các phiên để lựa chọn và chuyển đến phiên đầu tiên của bạn (`[0]`). Sau đó, hủy phiên `ssh` theo tên:

  Nhập `Ctrl` + `b` - `s`, sử dụng các phím mũi tên để đánh dấu phiên `0` và nhấn `enter` `tmux kill-session -t ssh`.

- Cuối cùng, tách khỏi phiên hiện tại và chấm dứt nó theo tên:

  `Ctrl` + `b` - `d` `tmux kill-session -t 0`.

Đáp án Bài tập Mở rộng

1. Cả `screen` và `tmux` đều có thể vào chế độ dòng lệnh thông qua *tiền tố lệnh* + `:` (chúng ta đã thấy một ví dụ ngắn gọn với `tmux`). Hãy tìm hiểu và thực hiện các tác vụ sau trong chế độ dòng lệnh:
   - Đưa `screen` vào chế độ sao chép:

     `Ctrl` + `a` - `:` — sau đó gõ `copy`.
   - Buộc `tmux` đổi tên cửa sổ hiện tại:

     `Ctrl` + `b` - `:` — sau đó gõ `rename-window`.
   - Buộc `screen` đóng tất cả các cửa sổ và kết thúc phiên:

     `Ctrl` + `a` - `:` — sau đó gõ `quit`.
   - Buộc `tmux` chia một khung thành hai:

     `Ctrl` + `b` - `:` — sau đó gõ `split-window`.
   - Buộc `tmux` chấm dứt cửa sổ hiện tại:

     `Ctrl` + `b` - `:` — sau đó gõ `kill-window`.
2. Khi vào chế độ sao chép trong `screen`, bạn không chỉ có thể sử dụng các phím mũi tên và `PgUP` hoặc `PgDown` để điều hướng cửa sổ hiện tại và bộ đệm cuộn lùi mà còn có khả năng sử dụng một trình chỉnh sửa toàn màn hình giống như vi. Bằng cách sử dụng trình chỉnh sửa này, hãy thực hiện các tác vụ sau:
   - Báo lại `supercalifragilisticexpialidocious` trong cửa sổ dòng lệnh `screen` của bạn:

     ```
     echo supercalifragilisticexpialidocious
     ```
   - Bây giờ, sao chép năm ký tự liên tiếp (từ trái sang phải) trong dòng ngay phía trên con trỏ của bạn:

     Chúng ta vào chế độ sao chép: `Ctrl` + `a` - `[` hoặc `Ctrl` + `a` - `:` rồi gõ `copy`. Sau đó, di chuyển đến dòng trên bằng cách sử dụng `k` và nhấn `Space` để đánh dấu việc bắt đầu vùng chọn. Cuối cùng, di chuyển về phía trước bốn ký tự bằng cách sử dụng `l` và nhấn lại `Space` để đánh dấu kết thúc vùng chọn.
   - Cuối cùng, dán vùng chọn (`stice`) trở lại vào dấu nhắc lệnh:

     `Ctrl` + `a` - `]`

3. Giả sử bạn muốn chia sẻ phiên `tmux` (`our_session`) với người dùng khác. Bạn đã tạo ổ nối (`/tmp/our_socket`) với các quyền phù hợp để cả bạn và người dùng khác đều có thể đọc và ghi được. Cần đáp ứng hai điều kiện nào khác để người dùng thứ hai có thể đính thành công phiên thông qua `tmux -S /tmp/our_socket a -t our_session`?

   Cả hai người dùng phải có một nhóm chung, ví dụ như `multiplexer`. Sau đó, bạn cũng sẽ phải thay đổi ổ nối theo nhóm đó: `chgrp multiplexer /tmp/our_socket`.

103.6 Sửa đổi Ưu tiên thực hiện Tiến trình

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 103.6

Khối lượng

2

Các lĩnh vực kiến thức chính

- Biết mức độ ưu tiên mặc định của công việc được tạo.
- Chạy một chương trình có mức độ ưu tiên cao hơn hoặc thấp hơn mặc định.
- Thay đổi mức độ ưu tiên của một tiến trình đang chạy.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

- `nice`
- `ps`
- `renice`
- `top`

103.6 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.6 Sửa đổi Ưu tiên thực hiện Tiến trình
Bài:	1 trên 1

Giới thiệu

Các hệ điều hành có thể chạy nhiều tiến trình cùng một lúc được gọi là hệ thống đa nhiệm hoặc đa xử lý. Trong khi tính đồng thời thực sự chỉ xảy ra khi có sẵn nhiều đơn vị xử lý thì ngay cả các hệ thống bộ xử lý đơn lẻ cũng có thể bắt chước được tính đồng thời này bằng cách chuyển đổi thật nhanh giữa các tiến trình. Kỹ thuật này cũng được sử dụng trong các hệ thống có nhiều CPU tương đương hoặc *hệ thống đa bộ xử lý đối xứng (SMP)*, với điều kiện là số lượng tiến trình đồng thời tiềm năng phải vượt xa số lượng đơn vị bộ xử lý có sẵn.

Trên thực tế, chỉ có một tiến trình có thể điều khiển CPU tại một thời điểm nhất định. Tuy nhiên, hầu hết các hoạt động của tiến trình là *các lời gọi hệ thống*, nghĩa là tiến trình đang chạy sẽ chuyển quyền kiểm soát CPU sang tiến trình của hệ điều hành để nó thực hiện thao tác được yêu cầu. Các phiên gọi hệ thống sẽ phụ trách tất cả các giao tiếp giữa các thiết bị như phân bổ bộ nhớ, đọc và ghi trên hệ thống tệp, in văn bản trên màn hình, tương tác người dùng, chuyển mạng, v.v. Việc chuyển quyền điều khiển CPU trong khi gọi hệ thống cho phép hệ điều hành quyết định sẽ trả lại quyền kiểm soát CPU cho tiến trình trước đó hay trao nó cho một tiến trình khác. Vì các CPU hiện đại có thể thực thi các lệnh nhanh hơn nhiều so với tốc độ giao tiếp của các phần cứng ngoại vi với nhau, một tiến trình kiểm soát mới có thể thực hiện nhiều công việc của CPU trong khi các phản hồi phần cứng được yêu cầu trước đó vẫn chưa khả dụng. Để đảm bảo khai thác tối đa CPU, các hệ

điều hành đa xử lý sẽ giữ một danh sách động gồm các tiến trình đang hoạt động chờ tới "lượt" để sử dụng CPU. Mặc dù chúng cho phép cải thiện đáng kể thời gian sử dụng CPU nhưng nếu chỉ dựa vào lời gọi hệ thống để chuyển đổi giữa các tiến trình thì hiệu suất đa nhiệm sẽ không đạt được một cách thỏa đáng. Một tiến trình không thực hiện lời gọi hệ thống nào có thể kiểm soát CPU vô thời hạn. Đây là lý do tại sao các hệ điều hành hiện đại cũng *mang tính ưu tiên*, nghĩa là một tiến trình đang chạy có thể được đưa trở lại hàng đợi để một tiến trình quan trọng hơn có thể kiểm soát CPU ngay cả khi tiến trình đang chạy đó chưa thực hiện một lời gọi hệ thống nào.

Trình Lập Lịch Biểu Linux

Với tư cách là một hệ điều hành đa xử lý mang tính ưu tiên, Linux có triển khai một trình lập lịch biểu (Linux Scheduler) để tổ chức hàng đợi dành cho các tiến trình. Chính xác hơn, trình lập lịch biểu cũng quyết định xem *luồng* đang xếp hàng đợi nào sẽ được thực thi — một tiến trình có thể phân nhánh ra nhiều luồng độc lập. Tuy nhiên, tiến trình và luồng là các thuật ngữ có thể hoán đổi cho nhau trong ngữ cảnh này. Mọi tiến trình đều có hai thuộc tính can thiệp vào việc lập lịch biểu của nó: *chính sách lập lịch biểu* và *ưu tiên lập lịch biểu*.

Có hai loại chính sách lập lịch biểu chính: *chính sách thời gian thực* và *chính sách thông thường*. Các tiến trình theo chính sách thời gian thực sẽ được lên lịch trực tiếp theo các giá trị ưu tiên của chúng. Nếu một tiến trình quan trọng hơn đã sẵn sàng chạy, một tiến trình đang chạy ít quan trọng hơn sẽ phải "nhường" và tiến trình có mức ưu tiên cao hơn sẽ kiểm soát CPU. Một tiến trình có mức ưu tiên thấp hơn sẽ chỉ giành được quyền kiểm soát CPU nếu các tiến trình có mức ưu tiên cao hơn không hoạt động hoặc đang chờ phản hồi của phần cứng.

Bất kỳ tiến trình thời gian thực nào cũng có mức độ ưu tiên cao hơn một tiến trình thông thường. Là một hệ điều hành có mục đích phổ thông, Linux chỉ chạy một vài tiến trình thời gian thực. Hầu hết các tiến trình (bao gồm cả chương trình hệ thống và người dùng) đều chạy theo các chính sách lập lịch biểu thông thường. Các tiến trình thông thường thường có cùng một giá trị ưu tiên, nhưng các chính sách thông thường có thể xác định các quy tắc ưu tiên thực thi bằng cách sử dụng một thuộc tính từ tiến trình khác là giá trị *nice*. Để tránh nhầm lẫn với các ưu tiên động bắt nguồn từ các giá trị nice, các ưu tiên lập lịch biểu thường được gọi là các ưu tiên lập lịch biểu *tĩnh*.

Trình lập lịch biểu Linux có thể được cấu hình theo nhiều cách khác nhau và thậm chí còn có nhiều cách phức tạp hơn để thiết lập mức độ ưu tiên, nhưng những khái niệm cơ bản này vẫn sẽ luôn được áp dụng. Khi kiểm tra và điều chỉnh tiến trình lập lịch biểu, điều quan trọng cần lưu ý là chỉ các tiến trình theo chính sách lập lịch biểu thông thường mới bị ảnh hưởng.

Ưu tiên Đọc

Linux dành các mức độ ưu tiên tĩnh từ 0 đến 99 cho các tiến trình thời gian thực và các tiến trình

bình thường được gán cho các mức độ ưu tiên tĩnh từ 100 đến 139, nghĩa là có 39 mức độ ưu tiên khác nhau cho các tiến trình thông thường. Giá trị thấp hơn có nghĩa là ưu tiên cao hơn. Mức độ ưu tiên tĩnh của một tiến trình đang hoạt động có thể được tìm thấy trong tệp `sched` nằm trong thư mục tương ứng của nó bên trong hệ thống tệp `/proc`:

```
$ grep ^prio /proc/1/sched
prio                  :       120
```

Như được minh họa trong ví dụ, dòng bắt đầu bằng `prio` đã đưa ra giá trị ưu tiên của tiến trình (tiến trình PID 1 là tiến trình *init* hoặc *systemd* - tiến trình đầu tiên mà hạt nhân khởi động trong quá trình khởi tạo hệ thống). Mức ưu tiên tiêu chuẩn cho các tiến trình thông thường là 120; do đó, nó có thể giảm xuống 100 hoặc tăng lên 139. Ta có thể xác minh mức độ ưu tiên của tất cả các tiến trình đang chạy bằng lệnh `ps -Al` hoặc `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
```

Cột `PRI` cho biết mức độ ưu tiên tĩnh được chỉ định bởi hạt nhân. Tuy nhiên, hãy lưu ý rằng giá trị ưu tiên được hiển thị bởi `ps` khác với giá trị thu được trong ví dụ trước. Vì nhiều lý do tiền lệ, mức độ ưu tiên được hiển thị theo `ps` sẽ nằm trong khoảng từ -40 đến 99 theo mặc định. Do đó, mức độ ưu tiên thực tế được tính bằng cách cộng 40 vào mức đó (cụ thể là 80 + 40 = 120).

Chúng ta cũng có thể theo dõi liên tục các tiến trình hiện đang được quản lý bởi nhân Linux bằng chương trình `top`. Như với `ps`, `top` cũng hiển thị giá trị ưu tiên theo một cách riêng. Để giúp xác định các tiến trình thời gian thực dễ dàng hơn, `top` sẽ trừ giá trị ưu tiên đi 100, từ đó làm cho tất cả các ưu tiên thời gian thực trở thành số âm với một số âm hoặc ký tự *rt* để xác định chúng. Do đó, mức độ ưu tiên thông thường được hiển thị theo `top` có phạm vi từ 0 đến 39.

NOTE

Để biết thêm chi tiết từ lệnh `ps`, ta có thể sử dụng các tùy chọn bổ sung. Hãy cùng so sánh đầu ra từ lệnh này với đầu ra từ ví dụ trước:

```
$ ps -e -o user,uid,comm,tty,pid,ppid,pri,pmem,pcpu --sort=-pcpu | head
```

Độ "Nice" của Quy Trình

Mỗi một tiến trình bình thường đều bắt đầu với giá trị nice mặc định là 0 (ưu tiên 120). Cái tên *nice* xuất phát từ ý tưởng rằng các tiến trình "tốt bụng hơn" sẽ cho phép các tiến trình khác chạy trước chúng trong một hàng đợi thực thi. Số nice nằm trong khoảng từ -20 (ít nice, ưu tiên cao) đến 19 (nice hơn, ưu tiên thấp). Linux cũng cho phép ta gán các giá trị nice khác nhau cho các luồng trong cùng một tiến trình. Cột `NI` trong đầu ra `ps` cho biết giá trị *nice* bằng số.

Chỉ siêu người dùng mới có thể giảm mức độ nice của tiến trình xuống dưới 0. Ta có thể bắt đầu một tiến trình với mức độ ưu tiên không chuẩn bằng lệnh `nice`. Theo mặc định, `nice` sẽ thay đổi mức độ thành 10, nhưng nó cũng có thể được chỉ định bằng tùy chọn `-n`:

```
$ nice -n 15 tar czf home_backup.tar.gz /home
```

Trong ví dụ này, lệnh `tar` được thực thi với độ chính xác là 15. Lệnh `renice` có thể được sử dụng để thay đổi mức độ ưu tiên của một tiến trình đang chạy. Tùy chọn `-p` sẽ cho biết số PID của tiến trình đích. Ví dụ:

```
# renice -10 -p 2164
2164 (process ID) old priority 0, new priority -10
```

Các tùy chọn `-g` và `-u` được sử dụng để sửa đổi tương ứng tất cả các tiến trình của một nhóm hoặc một người dùng cụ thể. Với `renice +5 -g users`, tính nice của các tiến trình do người dùng của nhóm *users* sở hữu sẽ được nâng lên năm lần.

Bên cạnh `renice`, mức độ ưu tiên của các tiến trình có thể được sửa đổi bằng các chương trình khác như `top`. Trên màn hình chính trên cùng, độ chuẩn của tiến trình có thể được sửa đổi bằng cách nhấn `r` và sau đó là số PID của tiến trình:

```
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
```

Thông báo `PID to renice [default pid = 1]` sẽ xuất hiện với tiến trình được liệt kê đầu tiên được chọn theo mặc định. Để thay đổi mức độ ưu tiên của một tiến trình khác, hãy nhập PID của tiến trình đó rồi nhấn Enter. Sau đó, thông báo `Renice PID 1 to value` sẽ xuất hiện (với số PID được yêu cầu) và ta có thể gán một giá trị nice mới.

Bài tập Hướng dẫn

1. Trong một hệ thống đa nhiệm mang tính ưu tiên, điều gì sẽ xảy ra khi một tiến trình có mức ưu tiên thấp hơn đang chiếm bộ xử lý và một tiến trình có mức ưu tiên cao hơn đang được xếp hàng để đợi được thực thi?

2. Hãy xem màn hình `top` sau:

```
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
```

   PID nào có ưu tiên thời gian thực?

3. Hãy xem danh sách `ps -el` sau:

```
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
```

PID nào có mức ưu tiên cao hơn?

4. Sau khi cố gắng đặt lại giá trị nice cho một tiến trình với `renice`, lỗi sau đã xảy ra:

```
$ renice -10 21704
renice: failed to set priority for 21704 (process ID): Permission denied
```

Nguyên nhân gây ra lỗi này là gì?

Bài tập Mở rộng

1. Việc thay đổi mức độ ưu tiên của tiến trình thường được yêu cầu khi một tiến trình chiếm quá nhiều thời gian của CPU. Khi sử dụng `ps` với các tùy chọn tiêu chuẩn để in tất cả các tiến trình hệ thống ở định dạng dài, cờ `--sort` nào sẽ sắp xếp các tiến trình theo mức sử dụng CPU theo thứ tự tăng dần?

2. Lệnh `schedtool` có khả năng đặt tất cả các tham số lập lịch biểu CPU mà Linux có khả năng thực thi hoặc hiển thị thông tin cho các tiến trình nhất định. Làm thế nào để nó có thể được sử dụng để hiển thị các tham số lập lịch biểu của tiến trình `1750`? Ngoài ra, làm cách nào để sử dụng `schedtool` để thay đổi tiến trình 1750 thành tiến trình thời gian thực với mức độ ưu tiên -90 (như được hiển thị bởi `top`)?

Tóm tắt

Bài học này đã trình bày cách Linux phân bổ thời gian của CPU cho các tiến trình nó quản lý. Để đảm bảo hiệu suất tốt nhất, các tiến trình quan trọng hơn phải vượt qua các tiến trình ít quan trọng hơn. Bài học đã đi qua các bước sau:

- Các khái niệm cơ bản về hệ thống đa xử lý.
- Trình lập lịch biểu tiến trình là gì và cách Linux triển khai nó.
- Ưu tiên của Linux là gì, giá trị "nice" và mục đích của chúng.
- Cách đọc và giải thích các ưu tiên của tiến trình trong Linux.
- Cách thay đổi mức độ ưu tiên của một tiến trình trước và trong khi thực hiện.

Đáp án Bài tập Hướng dẫn

1. Trong một hệ thống đa nhiệm mang tính ưu tiên, điều gì sẽ xảy ra khi một tiến trình có mức ưu tiên thấp hơn đang chiếm bộ xử lý và một tiến trình có mức ưu tiên cao hơn đang được xếp hàng để đợi được thực thi?

   Quá trình có mức ưu tiên thấp hơn sẽ tạm dừng và thay vào đó, quá trình có mức độ ưu tiên cao hơn sẽ được thực thi.

2. Hãy xem màn hình `top` sau:

```
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
```

   PID nào có ưu tiên thời gian thực?

   PIDs 12 và 16.

3. Hãy xem danh sách `ps -el` sau:

```
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
```

PID nào có mức ưu tiên cao hơn?

PID 12.

4. Sau khi cố gắng đặt lại giá trị nice cho một tiến trình với `renice`, lỗi sau đã xảy ra:

```
$ renice -10 21704
renice: failed to set priority for 21704 (process ID): Permission denied
```

Nguyên nhân gây ra lỗi này là gì?

Chỉ siêu người dùng mới có thể giảm các số chuẩn xuống dưới 0.

Đáp án Bài tập Mở rộng

1. Việc thay đổi mức độ ưu tiên của tiến trình thường được yêu cầu khi một tiến trình chiếm quá nhiều thời gian của CPU. Khi sử dụng `ps` với các tùy chọn tiêu chuẩn để in tất cả các tiến trình hệ thống ở định dạng dài, cờ `--sort` nào sẽ sắp xếp các tiến trình theo mức sử dụng CPU theo thứ tự tăng dần?

```
$ ps -el --sort=pcpu
```

2. Lệnh `schedtool` có khả năng đặt tất cả các tham số lập lịch biểu CPU mà Linux có khả năng thực thi hoặc hiển thị thông tin cho các tiến trình nhất định. Làm thế nào để nó có thể được sử dụng để hiển thị các tham số lập lịch biểu của tiến trình `1750`? Ngoài ra, làm cách nào để sử dụng `schedtool` để thay đổi tiến trình 1750 thành tiến trình thời gian thực với mức độ ưu tiên -90 (như được hiển thị bởi `top`)?

```
$ schedtool 1750
```

```
$ schedtool -R -p 89 1750
```

103.7 Tìm kiếm Tệp Văn bản bằng Biểu thức Chính quy

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 103.7

Khối lượng

3

Các lĩnh vực kiến thức chính

- Tạo các biểu thức chính quy đơn giản chứa một số phần tử ký hiệu.
- Hiểu sự khác biệt giữa các biểu thức chính quy cơ bản và mở rộng.
- Hiểu các khái niệm về ký tự đặc biệt, lớp ký tự, bộ định lượng và neo.
- Sử dụng các công cụ biểu thức chính quy để thực hiện tìm kiếm thông qua hệ thống tệp hoặc nội dung tệp.
- Sử dụng biểu thức thông thường để xóa, thay đổi và thay thế văn bản.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

- `grep`
- `egrep`
- `fgrep`
- `sed`
- `regex(7)`

103.7 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.7 Tìm kiếm Tệp Văn bản bằng Biểu thức Chính quy
Bài:	1 trên 2

Giới thiệu

Các thuật toán tìm kiếm theo chuỗi được sử dụng rộng rãi bởi nhiều tác vụ xử lý dữ liệu đến mức các hệ điều hành tương tự như Unix còn có một cách triển khai riêng đã trở nên vô cùng phổ biến là *Biểu thức chính quy*, thường được viết tắt là *REs* (Regular expressions). Biểu thức chính quy bao gồm các chuỗi ký tự tạo thành một mẫu chung được sử dụng để định vị và đôi khi là sửa đổi một chuỗi tương ứng trong một chuỗi ký tự lớn hơn. Biểu thức chính quy mở rộng đáng kể các khả năng sau:

- Viết quy tắc phân tích cú pháp cho các yêu cầu trong máy chủ HTTP, đặc biệt là *nginx*.
- Viết các tệp lệnh chuyển đổi bộ dữ liệu dựa trên văn bản sang một định dạng khác.
- Tìm kiếm các sự kiện được quan tâm trong các mục nhật ký hoặc tài liệu.
- Lọc tài liệu đánh dấu, giữ nội dung ngữ nghĩa.

Biểu thức chính quy đơn giản nhất chứa ít nhất một *nguyên tử*. Một nguyên tử (được đặt tên như vậy vì nó là thành phần cơ bản của một biểu thức chính quy) chỉ là một ký tự có thể có hoặc không có ý nghĩa gì đặc biệt. Hầu hết các ký tự thông thường đều khá rõ ràng và sẽ giữ nguyên nghĩa đen,

trong khi có một số ký tự có thể mang những ý nghĩa đặc biệt: `.` (dấu chấm):: Nguyên tử khớp với bất kỳ ký tự nào. `^` (dấu mũ):: Nguyên tử khớp với đầu dòng. `$` (dấu đô la):: Nguyên tử khớp với cuối dòng.

Ví dụ: biểu thức chính quy `bc` (được tạo bởi các nguyên tử chữ là `b` và `c`) có thể được tìm thấy trong chuỗi `abcd`, nhưng lại không thể được tìm thấy trong chuỗi `a1cd`. Mặt khác, ta có thể tìm thấy biểu thức chính quy `.c` trong cả hai chuỗi `abcd` và `a1cd`, vì dấu chấm `.` sẽ khớp với bất kỳ ký tự nào.

Các nguyên tử dấu mũ và ký hiệu đô la được sử dụng khi chúng ta chỉ quan tâm tới các kết quả trùng khớp ở đầu hoặc cuối chuỗi. Vì lý do đó, chúng còn được gọi là *neo*. Ví dụ: `cd` có thể được tìm thấy trong `abcd`, nhưng `^cd` thì không. Tương tự, `ab` có thể được tìm thấy trong `abcd`, nhưng `ab$` thì không. Dấu mũ `^` là một ký tự chữ trừ khi nó đứng ở đầu và `$` là ký tự chữ trừ khi nó đứng ở cuối biểu thức chính quy.

Biểu thức Ngoặc Vuông

Có một loại nguyên tử khác có tên là *biểu thức ngoặc vuông*. Mặc dù không phải là một ký tự đơn lẻ, dấu ngoặc vuông `[]` (bao gồm cả nội dung của chúng) vẫn được coi là một nguyên tử. Một biểu thức ngoặc vuông thường chỉ là một danh sách các ký tự chữ được đặt trong `[]` khiến cho nguyên tử khớp với bất kỳ ký tự đơn nào trong danh sách. Ví dụ: biểu thức `[1b]` có thể được tìm thấy trong cả hai chuỗi `abcd` và `a1cd`. Để chỉ định các ký tự mà nguyên tử không được tương ứng, danh sách phải bắt đầu bằng `^`, như trong `[^1b]`. Chúng ta cũng có thể chỉ định phạm vi ký tự trong biểu thức ngoặc. Ví dụ: `[0-9]` khớp với các chữ số từ 0 đến 9 và `[a-z]` khớp với bất kỳ chữ cái viết thường nào. Phạm vi phải được sử dụng một cách thận trọng vì chúng có thể sẽ không được nhất quán ở tất cả các khu vực.

Các danh sách biểu thức ngoặc vuông cũng chấp nhận các hạng (class) thay vì chỉ các ký tự và phạm vi đơn lẻ. Các hạng ký tự truyền thống là:

`[:alnum:]`
: Đại diện cho một ký tự chữ và số.

`[:alpha:]`
: Đại diện cho một ký tự chữ cái.

`[:ascii:]`
: Đại diện cho một ký tự phù hợp với bộ ký tự ASCII.

`[:blank:]`
: Đại diện cho một ký tự trống, nghĩa là một khoảng trắng hoặc một ký tự tab.

`[:cntrl:]`
Đại diện cho một ký tự điều khiển.

`[:digit:]`
Đại diện cho một chữ số (0 đến 9).

`[:graph:]`
Đại diện cho bất kỳ ký tự nào có thể in được ngoại trừ khoảng trắng.

`[:lower:]`
Đại diện cho một ký tự chữ thường.

`[:print:]`
Đại diện cho bất kỳ ký tự nào có thể in, kể cả khoảng trắng.

`[:punct:]`
Đại diện cho bất kỳ ký tự nào có thể in mà không phải là khoảng trắng hoặc ký tự chữ và số.

`[:space:]`
Đại diện cho các ký tự khoảng trắng: dấu cách, nguồn cấp dữ liệu (`\f`), xuống dòng (`\n`), dấu xuống dòng (`\r`), tab ngang (`\t`) và tab dọc (`\ v`).

`[:upper:]`
Đại diện cho một chữ hoa.

`[:xdigit:]`
Đại diện cho các chữ số thập lục phân (0 đến F).

Các hạng ký tự có thể được kết hợp với các ký tự và phạm vi đơn lẻ nhưng lại không thể được sử dụng làm điểm cuối cho một phạm vi. Ngoài ra, các hạng ký tự chỉ có thể được sử dụng trong các biểu thức ngoặc vuông chứ không phải là một nguyên tử độc lập bên ngoài dấu ngoặc.

Định lượng

Phạm vi tiếp cận của một nguyên tử, nguyên tử ký tự đơn hoặc nguyên tử ngoặc vuông có thể được điều chỉnh bằng cách sử dụng *bộ định lượng nguyên tử*. Bộ định lượng nguyên tử sẽ xác định các *chuỗi* nguyên tử - tức các kết quả trùng khớp xảy ra khi một lượt lặp lại liền kề của nguyên tử được tìm thấy trong chuỗi. Chuỗi con tương ứng với kết quả trùng khớp được gọi là các *mảnh*. Mặc dù vậy, bộ định lượng và các tính năng khác của biểu thức chính quy sẽ được xử lý khác nhau tùy thuộc vào tiêu chuẩn đang được sử dụng.

Theo định nghĩa của POSIX, có hai dạng biểu thức chính quy: biểu thức chính quy "cơ bản" và biểu thức chính quy "mở rộng". Hầu hết các chương trình liên quan đến văn bản trong bất kỳ bản phân phối Linux thông thường nào cũng đều hỗ trợ cả hai dạng này. Vì vậy, chúng ta phải biết được sự khác biệt giữa chúng để tránh các vấn đề về tương thích và để chọn cách triển khai phù hợp nhất cho tác vụ dự kiến.

Bộ định lượng `*` có cùng chức năng trong cả RE cơ bản và RE mở rộng (nguyên tử không xuất hiện hoặc xuất hiện nhiều lần) và nó sẽ là một ký tự chữ nếu xuất hiện ở đầu biểu thức chính quy hoặc trước dấu gạch chéo ngược `\`. Bộ định lượng dấu cộng `` ` `` sẽ chọn các mảnh có chứa một hoặc nhiều nguyên tử trùng khớp theo chuỗi. Với bộ định lượng dấu chấm hỏi `` `?` ``, một kết quả trùng khớp sẽ xảy ra nếu nguyên tử tương ứng xuất hiện một hoặc không lần. Nếu trước dấu gạch chéo ngược `` `\` ``, ý nghĩa đặc biệt của chúng sẽ không được xét. Các biểu thức chính quy cơ bản cũng hỗ trợ các bộ định lượng `` ` `` và `?` nhưng chúng cần được đặt trước dấu gạch chéo ngược. Không giống như các biểu thức chính quy mở rộng, bản thân `+` và `?` là các ký tự chữ trong các biểu thức chính quy cơ bản.

Giới hạn

Giới hạn là một bộ định lượng nguyên tử mà đúng như tên gọi, nó cho phép người dùng chỉ định ranh giới số lượng chính xác cho một nguyên tử. Trong các biểu thức chính quy mở rộng, một giới hạn có thể xuất hiện dưới ba dạng:

`{i}`
: Nguyên tử phải xuất hiện đúng `i` lần (`i` là một số nguyên). Ví dụ: `[[:blank:]]{2}` sẽ khớp với chính xác hai ký tự trống.

`{i,}`
: Nguyên tử phải xuất hiện ít nhất `i` lần (`i` là một số nguyên). Ví dụ: `[[:blank:]]{2,}` sẽ khớp với bất kỳ chuỗi nào gồm hai ký tự trống trở lên.

`{i,j}`
: Nguyên tử phải xuất hiện ít nhất `i` lần và nhiều nhất `j` lần (`i` và `j` là các số nguyên, `j` lớn hơn `i`). Ví dụ: `xyz{2,4}` sẽ khớp với chuỗi `xy` theo sau là hai đến bốn ký tự `z`.

Trong mọi trường hợp, nếu một chuỗi con khớp với một biểu thức chính quy và một chuỗi con dài hơn bắt đầu từ cùng một điểm cũng khớp thì chuỗi con dài hơn sẽ được xét.

Các biểu thức chính quy cơ bản cũng hỗ trợ các giới hạn, nhưng các dấu phân cách phải được đặt trước `\`: `\{` và `\}`. Bản thân `{` và `}` được hiểu là các ký tự chữ. Một ký tự `\{` được theo sau bởi một ký tự số sẽ được coi là một ký tự chữ chứ không phải là phần đầu của một giới hạn.

Nhánh và Tham chiếu Ngược

Biểu thức chính quy cơ bản cũng khác với biểu thức chính quy mở rộng ở một khía cạnh quan trọng khác: một biểu thức chính quy mở rộng có thể được chia thành nhiều *nhánh*, mỗi nhánh là một biểu thức chính quy độc lập. Các nhánh được phân tách bằng `|` và biểu thức chính quy kết hợp sẽ khớp với bất kỳ mẫu nào tương ứng với bất kỳ nhánh nào. Ví dụ: `he|him` sẽ khớp nếu một trong hai chuỗi con `he` hoặc `him` được tìm thấy trong chuỗi đang được kiểm tra. Các biểu thức chính quy cơ bản diễn giải `|` như một ký tự chữ. Tuy nhiên, hầu hết các chương trình hỗ trợ các biểu thức chính quy cơ bản sẽ cho phép các nhánh có chứa `\|`.

Một biểu thức chính quy mở rộng được đặt trong `()` có thể được sử dụng trong *tham chiếu ngược*. Ví dụ: `([[:digit:]])\1` sẽ khớp với bất kỳ biểu thức chính quy nào lặp lại chính nó ít nhất một lần, bởi vì `\1` trong biểu thức là tham chiếu ngược tới mảnh được khớp bởi biểu thức con trong dấu ngoặc đơn đầu tiên. Nếu có nhiều hơn một biểu thức con được đặt trong ngoặc đơn tồn tại trong biểu thức chính quy, chúng có thể được tham chiếu bằng `\2`, `\3`, v.v.

Đối với các RE cơ bản, các biểu thức con phải được bao quanh bởi `\(` và `\)` (với bản thân `(` và `)` được coi là các ký tự thông thường). Chỉ số tham chiếu ngược được sử dụng giống như trong các biểu thức chính quy mở rộng.

Tìm kiếm với Biểu thức chính quy

Lợi ích rõ ràng nhất mà biểu thức chính quy mang lại là sự cải thiện trong việc tìm kiếm trên hệ thống tệp và trong tài liệu văn bản. Tùy chọn `-regex` của lệnh `find` sẽ cho phép ta kiểm tra mọi đường dẫn trong hệ thống phân cấp thư mục dựa trên biểu thức chính quy. Ví dụ,

```
$ find $HOME -regex '.*/\..*' -size +100M
```

sẽ tìm kiếm các tệp lớn hơn 100 megabyte (100 đơn vị 1048576 byte) nhưng chỉ trong các đường dẫn bên trong thư mục chính của người dùng và có chứa kết quả trùng khớp với `.*/\..*`, tức là có `/.` bao quanh bởi bất kỳ một số lượng ký tự nào khác. Nói cách khác, chỉ các tệp ẩn hoặc tệp bên trong các thư mục ẩn mới được liệt kê, bất kể vị trí của `/.` là ở đâu trong đường dẫn tương ứng. Thay vào đó, đối với các biểu thức chính quy không phân biệt chữ hoa chữ thường, ta nên sử dụng tùy chọn `-iregex`:

```
$ 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
```

Trong ví dụ này, biểu thức chính quy chứa các nhánh (được viết theo kiểu *mở rộng*) để chỉ liệt kê các tệp phông chữ cụ thể trong phân cấp thư mục `/usr/share/fonts`. Biểu thức chính quy mở rộng không được hỗ trợ theo mặc định nhưng `find` sẽ cho phép kích hoạt chúng với `-regextype posix-extended` hoặc `-regextype egrep`. Tiêu chuẩn RE mặc định cho `find` là *findutils-default* - gần như là một bản sao của biểu thức chính quy cơ bản.

Thông thường, ta sẽ cần phải truyền đầu ra của chương trình cho lệnh `less` khi nó không vừa với màn hình. Lệnh `less` sẽ chia đầu vào của nó thành các trang, mỗi trang đều sẽ lấp đầy một màn hình, cho phép người dùng dễ dàng điều hướng văn bản lên và xuống. Ngoài ra, `less` cũng cho phép người dùng thực hiện các tìm kiếm dựa trên biểu thức chính quy. Tính năng này đặc biệt quan trọng vì `less` là trình phân trang mặc định được sử dụng cho nhiều tác vụ hàng ngày, chẳng hạn như kiểm tra các mục nhật ký hoặc tra cứu các trang hướng dẫn sử dụng. Ví dụ, khi đọc một trang hướng dẫn, việc nhấn phím `/` sẽ mở ra một dấu nhắc tìm kiếm. Đây là một tình huống điển hình mà trong đó, các biểu thức chính quy rất hữu ích vì các tùy chọn lệnh sẽ được liệt kê ngay sau lề trang trong bố cục trang hướng dẫn chung. Tuy nhiên, cùng một tùy chọn có thể xuất hiện nhiều lần trong văn bản, khiến cho việc tìm kiếm theo nghĩa đen trở nên không khả thi. Dù vậy, việc gõ `^[[:blank:]]*-o` — hoặc đơn giản hơn là `^ *-o` — trong dấu nhắc tìm kiếm vẫn sẽ nhảy ngay đến tùy chọn phần `-o` (nếu nó tồn tại) sau khi nhấn Enter, từ đó cho phép người dùng tham khảo mô tả của tùy chọn nhanh hơn.

Bài tập Hướng dẫn

1. Biểu thức chính quy mở rộng nào sẽ khớp với tất cả các địa chỉ email, chẳng hạn như `info@example.org`?

2. Biểu thức chính quy mở rộng nào sẽ chỉ khớp với mọi địa chỉ IPv4 ở định dạng tứ giác chấm tiêu chuẩn (như `192.168.15.1)?

3. Làm cách nào để có thể sử dụng lệnh `grep` để liệt kê nội dung của tệp `/etc/services` và loại bỏ tất cả các chú thích (dòng bắt đầu bằng #)?

4. Tệp `domains.txt` chứa danh sách các tên miền, mỗi dòng dành cho một tên miền. Lệnh `egrep` sẽ được sử dụng như thế nào để chỉ liệt kê các miền `.org` hoặc `.com`?

Bài tập Mở rộng

1. Từ thư mục hiện tại, lệnh `find` sẽ sử dụng biểu thức chính quy mở rộng như thế nào để tìm kiếm tất cả các tệp không chứa hậu tố tệp tiêu chuẩn (ví dụ: tên tệp không kết thúc bằng `.txt` hoặc `.c`)?

2. Lệnh `less` là lệnh phân trang mặc định để hiển thị các tệp văn bản dài trong môi trường vỏ. Bằng cách nhập `/`, một biểu thức chính quy có thể được nhập vào dấu nhắc tìm kiếm để chuyển đến kết quả trùng khớp tương ứng đầu tiên. Để giữ nguyên vị trí tài liệu hiện tại và chỉ đánh dấu các kết quả trùng khớp tương ứng, tổ hợp phím nào cần được nhập tại dấu nhắc tìm kiếm?

3. Trong `less`, làm cách nào để có thể lọc đầu ra sao cho chỉ những dòng khớp với biểu thức chính quy mới được hiển thị?

Tóm tắt

Bài học này đã đề cập đến sự hỗ trợ tổng quát của Linux về các biểu thức chính quy - một tiêu chuẩn được sử dụng rộng rãi có khả năng so khớp mẫu được hỗ trợ bởi hầu hết các chương trình liên quan đến văn bản. Bài học đã đi qua các bước sau:

- Biểu thức chính quy là gì.
- Các thành phần chính của một biểu thức chính quy.
- Sự khác nhau giữa biểu thức chính quy thông thường và biểu thức chính quy mở rộng.
- Cách thực hiện tìm kiếm văn bản và tệp đơn giản bằng các biểu thức chính quy.

Đáp án Bài tập Hướng dẫn

1. Biểu thức chính quy mở rộng nào sẽ khớp với tất cả các địa chỉ email, chẳng hạn như `info@example.org`?

```
egrep "\S+@\S+\.\S+"
```

2. Biểu thức chính quy mở rộng nào sẽ chỉ khớp với mọi địa chỉ IPv4 ở định dạng tứ giác chấm tiêu chuẩn (như `192.168.15.1)?

```
egrep "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
```

3. Làm cách nào để có thể sử dụng lệnh `grep` để liệt kê nội dung của tệp `/etc/services` và loại bỏ tất cả các chú thích (dòng bắt đầu bằng #)?

```
grep -v ^# /etc/services
```

4. Tệp `domains.txt` chứa danh sách các tên miền, mỗi dòng dành cho một tên miền. Lệnh `egrep` sẽ được sử dụng như thế nào để chỉ liệt kê các miền `.org` hoặc `.com`?

```
egrep ".org$|.com$" domains.txt
```

Đáp án Bài tập Mở rộng

1. Từ thư mục hiện tại, lệnh `find` sẽ sử dụng biểu thức chính quy mở rộng như thế nào để tìm kiếm tất cả các tệp không chứa hậu tố tệp tiêu chuẩn (ví dụ: tên tệp không kết thúc bằng `.txt` hoặc `.c`)?

   ```
   find . -type f -regextype egrep -not -regex '.*\.[[:alnum:]]{1,}$'
   ```

2. Lệnh `less` là lệnh phân trang mặc định để hiển thị các tệp văn bản dài trong môi trường vỏ. Bằng cách nhập `/`, một biểu thức chính quy có thể được nhập vào dấu nhắc tìm kiếm để chuyển đến kết quả trùng khớp tương ứng đầu tiên. Để giữ nguyên vị trí tài liệu hiện tại và chỉ đánh dấu các kết quả trùng khớp tương ứng, tổ hợp phím nào cần được nhập tại dấu nhắc tìm kiếm?

   Nhấn `Ctrl` + `K` trước khi nhập biểu thức tìm kiếm.

3. Trong `less`, làm cách nào để có thể lọc đầu ra sao cho chỉ những dòng khớp với biểu thức chính quy mới được hiển thị?

   Bằng cách nhấn `&` và nhập biểu thức tìm kiếm.

Linux Professional Institute

103.7 Bài 2

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.7 Tìm kiếm Tệp Văn bản bằng Biểu thức Chính quy
Bài:	2 trên 2

Giới thiệu

Việc truyền dữ liệu qua một chuỗi các lệnh được dẫn ống cho phép chúng ta áp dụng các bộ lọc phức hợp dựa trên các biểu thức chính quy. Biểu thức chính quy là một kỹ thuật quan trọng không chỉ được sử dụng trong quản trị hệ thống mà còn trong cả việc *khai phá dữ liệu* và các lĩnh vực liên quan. Hai lệnh đặc biệt phù hợp để thao tác với tệp và dữ liệu văn bản bằng biểu thức chính quy là `grep` và `sed`. `grep` là công cụ tìm mẫu và `sed` là trình chỉnh sửa luồng. Bản thân chúng đã rất hữu ích, nhưng chúng thậm chí sẽ còn nổi bật hơn khi được sử dụng cùng với các tiến trình khác.

Công cụ tìm Mẫu: grep

Một trong những cách sử dụng phổ biến nhất của `grep` là tạo điều kiện cho việc kiểm tra các tệp dài sử dụng biểu thức chính quy làm bộ lọc được áp dụng cho mỗi dòng. Nó có thể được sử dụng để chỉ hiển thị các dòng bắt đầu bằng một thuật ngữ nhất định. Ví dụ: `grep` có thể được sử dụng để kiểm tra một tệp cấu hình cho các mô-đun nhân và chỉ liệt kê các dòng tùy chọn:

```
$ 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
```

Ký tự ống `|` có thể được sử dụng để chuyển hướng trực tiếp đầu ra của lệnh sang đầu vào của `grep`. Ví dụ sau sử dụng biểu thức ngoặc vuông để chọn các dòng từ đầu ra `fdisk -l` bắt đầu bằng `Disk /dev/sda` hoặc `Disk /dev/sdb`:

```
# fdisk -l | grep '^Disk /dev/sd[ab]'
Disk /dev/sda: 320.1 GB, 320072933376 bytes, 625142448 sectors
Disk /dev/sdb: 7998 MB, 7998537728 bytes, 15622144 sectors
```

Việc chỉ lựa chọn các dòng có kết quả trùng khớp có thể sẽ không phù hợp với một số tác vụ cụ thể đòi hỏi phải điều chỉnh hành vi của `grep` thông qua các tùy chọn của nó. Ví dụ: tùy chọn `-c` hoặc `--count` sẽ cho `grep` biết có bao nhiêu dòng phù hợp:

```
# fdisk -l | grep '^Disk /dev/sd[ab]' -c
2
```

Tùy chọn có thể được đặt trước hoặc sau biểu thức chính quy. Các tùy chọn `grep` quan trọng khác là:

`-c` hoặc `--count`

Thay vì hiển thị kết quả tìm kiếm, nó sẽ chỉ hiển thị tổng số lần trùng khớp xảy ra trong bất kỳ tệp cụ thể nào.

`-i` hoặc `--ignore-case`

Khiến cho phiên tìm kiếm không phân biệt chữ hoa chữ thường.

`-f FILE` hoặc `--file=FILE`

Cho biết tệp chứa biểu thức chính quy sẽ sử dụng.

`-n` hoặc `--line-number`

Hiển thị số dòng.

`-v` hoặc `--invert-match`

Chọn mọi dòng ngoại trừ những dòng có chứa kết quả trùng khớp.

`-H` hoặc `--with-filename`

In cả tên của tệp có chứa dòng.

`-z` hoặc `--null-data`

Thay vì để `grep` xử lý các luồng dữ liệu đầu vào và đầu ra dưới dạng các dòng riêng biệt (sử dụng ký tự *xuống dòng* theo mặc định), lệnh sẽ lấy đầu vào hoặc đầu ra dưới dạng một chuỗi các dòng. Khi kết hợp đầu ra từ lệnh `find` bằng tùy chọn `-print0` với lệnh `grep`, tùy chọn `-z` hoặc `--null-data` nên được sử dụng để xử lý luồng theo cách tương tự.

Mặc dù được kích hoạt mặc định khi nhiều đường dẫn tệp được cung cấp làm đầu vào, tùy chọn `-H` sẽ không được kích hoạt cho các tệp đơn lẻ. Điều đó có thể sẽ rất quan trọng trong các tình huống đặc biệt, chẳng hạn như khi `grep` được gọi trực tiếp bởi `find`:

```
$ 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
```

Trong ví dụ này, `find` sẽ liệt kê mọi tệp trong `/usr/share/doc`, sau đó chuyển từng tệp tới `grep`, từ đó thực hiện tìm kiếm `3d moedeling` không phân biệt chữ hoa chữ thường bên trong tệp. Đường ống dẫn tới `cut` ở đó chỉ để giới hạn độ dài đầu ra ở 100 cột. Tuy nhiên, hãy lưu ý rằng không có cách nào để biết các dòng này đến từ tệp nào. Vấn đề này sẽ được giải quyết bằng cách thêm `-H` vào `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
```

Giờ đây, ta có thể xác định các tệp nơi mà mỗi kết quả trùng khớp được tìm thấy. Để làm cho danh sách trở nên đầy đủ hơn, các dòng đầu và cuối có thể được thêm vào các dòng có kết quả trùng khớp:

```
$ 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.
```

Tùy chọn `-1` sẽ hướng dẫn `grep` bao gồm thêm một dòng trước và sau khi nó tìm thấy một dòng có kết quả trùng khớp. Những dòng bổ sung này được gọi là *dòng ngữ cảnh* và được xác định trong đầu ra bằng dấu trừ sau tên tệp. Ta có thể thu được kết quả tương tự với `-C 1` hoặc `--context=1` và các số lượng dòng ngữ cảnh khác có thể được chỉ định.

Có hai chương trình bổ trợ cho grep là `egrep` và `fgrep`. Chương trình `egrep` tương đương với lệnh `grep -E` có kết hợp các tính năng bổ sung ngoài các biểu thức chính quy cơ bản. Ví dụ: với `egrep`, ta có thể sử dụng các tính năng biểu thức chính quy mở rộng như phân nhánh:

```
$ 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.
```

Trong ví dụ này, `3D modeling` hoặc `3D printing` sẽ khớp với biểu thức và không phân biệt chữ hoa chữ thường. Để chỉ hiển thị các phần của luồng văn bản khớp với biểu thức được sử dụng bởi `egrep`, hãy sử dụng tùy chọn `-o`.

Chương trình `fgrep` cũng tương đương với `grep -F`, tức là nó không phân tích cú pháp các biểu thức chính quy. Nó rất hữu ích trong các phiên tìm kiếm đơn giản mà mục tiêu là khớp với một biểu thức bằng chữ. Do đó, các ký tự đặc biệt như ký hiệu đô la và dấu chấm sẽ được hiểu theo nghĩa đen chứ không theo nghĩa của chúng trong biểu thức chính quy.

Trình chỉnh sửa Luồng: sed

Mục đích của chương trình `sed` là sửa đổi dữ liệu dựa trên văn bản theo một cách không tương tác. Có nghĩa là tất cả các thao tác chỉnh sửa đều sẽ được thực hiện theo hướng dẫn định sẵn chứ không phải tùy tiện gõ trực tiếp vào văn bản hiển thị trên màn hình. Theo thuật ngữ hiện đại, `sed` có thể được hiểu là trình phân tích cú pháp mẫu: cho một văn bản làm đầu vào, nó sẽ đặt nội dung tùy chỉnh tại các vị trí được xác định trước hoặc khi nó tìm thấy kết quả trùng khớp cho một biểu thức chính quy.

Sed, như tên của nó, rất phù hợp với văn bản được truyền qua các đường ống. Cú pháp cơ bản của nó là `sed -f SCRIPT` khi các hướng dẫn chỉnh sửa được lưu trữ trong tệp `SCRIPT` hoặc `sed -e COMMANDS` để thực thi `COMMANDS` trực tiếp từ dòng lệnh. Nếu không có `-f` hoặc `-e`, `sed` sẽ sử dụng tham số không phải tùy chọn đầu tiên làm tệp tệp lệnh. Ta cũng có thể sử dụng tệp làm đầu vào bằng cách cung cấp đường dẫn của nó làm đối số cho `sed`.

Các lệnh `sed` sẽ bao gồm một ký tự đơn, có thể đứng trước một địa chỉ hoặc theo sau một hoặc nhiều tùy chọn và sẽ được áp dụng mỗi lần cho một dòng. Địa chỉ có thể là một số dòng đơn, một biểu thức chính quy hoặc một phạm vi các dòng. Ví dụ: dòng đầu tiên của luồng văn bản có thể bị xóa bằng `1d`, trong đó, `1` chỉ định dòng mà lệnh xóa `d` sẽ được áp dụng. Để làm rõ cách sử dụng của `sed`, hãy lấy đầu ra của lệnh `factor `seq 12`` trả về các thừa số nguyên tố cho các số từ 1 đến 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
```

Việc xóa dòng đầu tiên bằng `sed` có thể được thực hiện bằng `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
```

Ta có thể chỉ định một loạt các dòng được ngăn cách bằng dấu phẩy:

```
$ factor `seq 12` | sed 1,7d
8: 2 2 2
9: 3 3
10: 2 5
11: 11
12: 2 2 3
```

Chúng ta có thể sử dụng nhiều hơn một lệnh được phân tách bằng dấu chấm phẩy trong cùng một lần thực thi. Tuy nhiên, trong trường hợp này, chúng ta phải đặt chúng trong dấu ngoặc đơn để vỏ không diễn giải dấu chấm phẩy:

```
$ factor `seq 12` | sed "1,7d;11d"
8: 2 2 2
9: 3 3
10: 2 5
12: 2 2 3
```

Trong ví dụ này, hai lệnh xóa đã được thực hiện, đầu tiên là trên các dòng từ 1 đến 7 và sau đó trên dòng 11. Một địa chỉ cũng có thể là một biểu thức chính quy; vì vậy, chỉ những dòng khớp mới bị ảnh hưởng bởi lệnh:

```
$ factor `seq 12` | sed "1d;/:.*2.*/d"
3: 3
```

```
5: 5
7: 7
9: 3 3
11: 11
```

Biểu thức chính quy `:.*2.*` sẽ khớp với bất kỳ lần xuất hiện nào của số 2 ở bất kỳ đâu sau dấu hai chấm và từ đó xóa đi các dòng tương ứng có chứa thừa số 2. Với `sed`, bất kỳ thứ gì được đặt giữa các dấu gạch chéo (`/`) đều được coi là biểu thức chính quy và theo mặc định, tất cả các RE cơ bản đều sẽ được hỗ trợ. Ví dụ: `sed -e "/^#/d" /etc/services` sẽ hiển thị nội dung của tệp `/etc/services` mà không có dòng bắt đầu bằng `#` (dòng chú thích).

Lệnh xóa `d` chỉ là một trong nhiều lệnh chỉnh sửa do `sed` cung cấp. Thay vì xóa một dòng, `sed` có thể thay thế nó bằng một đoạn văn bản nhất định:

```
$ 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
```

Hướng dẫn `c REMOVED` sẽ chỉ đơn giản là thay thế một dòng bằng `REMOVED`. Trong trường hợp của ví dụ, mọi dòng có chuỗi con khớp với biểu thức chính quy `:.*2.*` đều sẽ bị ảnh hưởng bởi lệnh `c REMOVED`. Lệnh `a TEXT` sẽ sao chép văn bản được chỉ định bởi `TEXT` sang một dòng mới ở sau dòng có kết quả trùng khớp. Lệnh `r FILE` cũng sẽ thực hiện tương tự nhưng sẽ sao chép nội dung của tệp được chỉ định bởi `FILE`. Lệnh `w` sẽ thực hiện ngược lại lệnh `r`, tức là dòng sẽ được thêm vào tệp được chỉ định.

Cho đến nay, lệnh `sed` được sử dụng nhiều nhất là `s/FIND/REPLACE/`. Lệnh này được sử dụng để thay thế kết quả trùng khớp với biểu thức chính quy `FIND` bằng văn bản được biểu thị bằng `REPLACE`. Ví dụ: lệnh `s/hda/sda/` sẽ thay thế một chuỗi con khớp với RE `hda` bằng `sda`. Chỉ kết quả trùng khớp đầu tiên được tìm thấy trong dòng sẽ được thay thế trừ khi cờ `g` được đặt sau lệnh như trong `s/hda/sda/g`.

Một ví dụ điển hình thực tế hơn sẽ giúp minh họa các tính năng của `sed`. Giả sử một phòng khám

y tế muốn gửi tin nhắn văn bản cho khách hàng của mình để nhắc nhở họ về lịch hẹn cho ngày hôm sau. Đây là một kịch bản triển khai điển hình dựa trên dịch vụ tin nhắn tức thời chuyên nghiệp. Dịch vụ này sẽ cung cấp API để truy cập vào hệ thống chịu trách nhiệm gửi tin nhắn. Những tin nhắn này thường bắt nguồn từ cùng một hệ thống chạy ứng dụng kiểm soát các cuộc hẹn với khách hàng và được kích hoạt bởi một thời điểm cụ thể trong ngày hoặc một số sự kiện khác. Trong tình huống giả định này, ứng dụng có thể tạo một tệp có tên `appointments.csv` chứa dữ liệu dạng bảng với tất cả các cuộc hẹn cho ngày hôm sau, sau đó được `sed` sử dụng để hiển thị tin nhắn văn bản từ tệp mẫu có tên `template.txt` . Các tệp CSV là một cách tiêu chuẩn để xuất dữ liệu từ các truy vấn cơ sở dữ liệu. Do đó, các cuộc hẹn mẫu có thể được đưa ra dưới dạng như sau:

```
$ cat appointments.csv
"NAME","TIME","PHONE"
"Carol","11am","55557777"
"Dave","2pm","33334444"
```

Dòng đầu tiên chứa các nhãn cho mỗi cột và sẽ được sử dụng để khớp với các thẻ bên trong tệp mẫu:

```
$ cat template.txt
Hey <NAME>, don't forget your appointment tomorrow at <TIME>.
```

Các dấu nhỏ hơn `<` và lớn hơn `>` được đặt xung quanh nhãn chỉ để giúp xác định chúng là thẻ. Tệp lệnh Bash sau đây phân tích cú pháp tất cả các cuộc hẹn được xếp hàng đợi bằng cách sử dụng `template.txt` làm mẫu tin nhắn:

```
#! /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
```

Một tệp lệnh sản xuất thực tế cũng sẽ xử lý tác vụ xác thực, kiểm tra lỗi và ghi nhật ký, nhưng ví dụ của chúng ta từ đầu đã có các chức năng cơ bản. Các lệnh đầu tiên được thực thi bởi `sed` chỉ được áp dụng cho dòng đầu tiên — địa chỉ 1 trong `1s/^"//;1s/","/\n/g;1s/"$//p` — nhằm xóa các dấu trích dẫn kép ở đầu và cuối — `1s/^"//` và `1s/"$//` — và để thay thế các dấu phân tách trường bằng một ký tự xuống dòng (`1s/","/\n/ g`). Chỉ có dòng đầu tiên là cần để tải tên cột; vì vậy, các dòng không khớp sẽ được nén bởi tùy chọn `-n` và yêu cầu ta đặt cờ `p` sau lệnh `sed` cuối cùng để in dòng khớp. Các thẻ sau đó sẽ được lưu trữ trong biến `TAGS` dưới dạng một mảng Bash. Một biến mảng Bash khác sẽ được tạo bởi lệnh `mapfile` để lưu các dòng chứa các cuộc hẹn trong biến mảng `ROWS`.

Vòng lặp `for` được sử dụng để xử lý từng dòng lịch hẹn được tìm thấy trong `ROWS`. Sau đó, dấu trích dẫn kép và dấu phân cách trong cuộc hẹn — cuộc hẹn nằm trong biến `${ROWS[$r]}` được sử dụng làm *here string* — sẽ được thay thế bằng `sed` tương tự như các lệnh được sử dụng để tải thẻ. Các giá trị riêng biệt cho cuộc hẹn sau đó sẽ được lưu trữ trong biến mảng `VALS` mà trong đó, các chỉ số con của mảng 0, 1 và 2 tương ứng với các giá trị cho `NAME`, `TIME` và `PHONE`.

Cuối cùng, một vòng lặp `for` lồng nhau sẽ đi qua mảng `TAGS` và thay thế từng thẻ được tìm thấy trong mẫu bằng giá trị tương ứng của nó trong `VALS`. Biến `MSG` sẽ giữ một bản sao của mẫu được hiển thị và được cập nhật bởi trình thay thế lệnh `s/<${TAGS[$c]}>/${VALS[$c]}/g` trên mỗi vòng lặp đi qua `TAGS`.

Điều này dẫn đến một thông báo được hiển thị như sau: `"Hey Carol, don’t forget your appointment tomorrow at 11am."` Sau đó, thông báo được hiển thị có thể được gửi dưới dạng tham số thông qua yêu cầu HTTP với `curl` dưới dạng thư hoặc bất kỳ phương pháp tương tự nào khác.

Kết hợp grep và sed

Các lệnh `grep` và `sed` có thể được sử dụng cùng nhau khi chúng ta cần các thủ tục khai phá văn bản phức tạp hơn. Ví dụ, với tư cách là quản trị viên hệ thống, chúng ta có thể sẽ phải kiểm tra tất cả các lần đăng nhập vào máy chủ. Tệp `/var/log/wtmp` ghi lại tất cả các lần đăng nhập và đăng xuất, trong khi tệp `/var/log/btmp` ghi lại các lần đăng nhập không thành công. Chúng được viết ở định dạng nhị phân và có thể được đọc bằng lệnh `last` và `lastb` tương ứng.

Đầu ra của `lastb` không chỉ hiển thị tên người dùng được sử dụng trong lần đăng nhập không hợp lệ mà còn cả địa chỉ IP của người dùng đó:

```
# 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
```

Tùy chọn `-d` sẽ dịch số IP thành tên máy chủ tương ứng. Tên máy chủ có thể cho ta biết về ISP hoặc dịch vụ lưu trữ được sử dụng để thực hiện những phiên đăng nhập không hợp lệ này. Tùy chọn `-a` sẽ đặt tên máy chủ trong cột cuối cùng, tạo điều kiện cho việc lọc chưa được áp dụng. Tùy chọn `--time-format notime` sẽ chặn thời gian khi có người cố đăng nhập. Lệnh `lastb` có thể sẽ mất một chút thời gian để hoàn thành nếu có quá nhiều lần đăng nhập không thành công. Do đó, đầu ra đã bị giới hạn ở mười mục nhập với tùy chọn `-n 10`.

Không phải IP từ xa nào cũng có một tên máy chủ liên kết. Vì vậy, DNS ngược sẽ không áp dụng cho chúng và chúng có thể bị loại bỏ. Mặc dù ta có thể viết một biểu thức chính quy để khớp với định dạng dự kiến cho tên máy chủ ở cuối dòng, nhưng có lẽ sẽ đơn giản hơn nếu ta viết một biểu thức chính quy để khớp với một chữ cái trong bảng chữ cái hoặc với một chữ số duy nhất ở cuối dòng. Ví dụ sau đây sẽ cho thấy cách lệnh `grep` lấy danh sách ở đầu vào tiêu chuẩn của nó và xóa các dòng không có tên máy chủ:

```
# 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
```

Lệnh `grep` với tùy chọn `-v` sẽ chỉ hiển thị các dòng không khớp với biểu thức chính quy đã cho. Một biểu thức chính quy khớp với bất kỳ dòng nào kết thúc bằng một số (ví dụ: `[0-9]$`) sẽ chỉ ghi lại các mục nhập không có tên máy chủ. Do đó, `grep -v '[0-9]$'` sẽ chỉ hiển thị các dòng kết thúc bằng tên máy chủ.

Đầu ra có thể được lọc kĩ hơn nữa bằng cách chỉ giữ lại tên miền và loại bỏ các phần khác khỏi mỗi dòng. Lệnh `sed` có thể thực hiện điều này bằng một trình thay thế lệnh để thay thế toàn bộ

dòng bằng tham chiếu ngược đến tên miền trong đó:

```
# 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
```

Dấu ngoặc đơn thoát trong `.* \(.*\)$` sẽ nhắc cho `sed` nhớ về phần giữa ký tự khoảng trắng cuối cùng và phần cuối của dòng. Trong ví dụ này, phần này được tham chiếu với `\1` và được sử dụng để thay thế toàn bộ dòng.

Rõ ràng là hầu hết các máy chủ từ xa đều đã cố gắng đăng nhập nhiều lần, vì thế mà tên miền đã lặp lại. Để chặn các mục nhập lặp lại, trước tiên, chúng cần được sắp xếp (bằng lệnh `sort`), sau đó chuyển qua lệnh `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
```

Điều này cho ta thấy cách kết hợp các lệnh khác nhau để tạo ra kết quả như mong muốn. Sau đó, danh sách tên máy chủ có thể được sử dụng để viết các quy tắc tường lửa hoặc thực hiện các biện pháp khác để thực thi bảo mật an ninh cho máy chủ.

Bài tập Hướng dẫn

1. Lệnh `last` sẽ hiển thị danh sách người dùng đăng nhập lần cuối, bao gồm cả IP gốc của họ. Lệnh `egrep` sẽ được sử dụng như thế nào để lọc đầu ra `last` sao cho chỉ hiển thị các lần xuất hiện của địa chỉ IPv4 và loại bỏ mọi thông tin bổ sung trong dòng tương ứng?

2. Tùy chọn nào nên được cung cấp cho `grep` để lọc chính xác đầu ra được tạo bởi lệnh `find` được thực thi với tùy chọn `-print0`?

3. Lệnh `uptime -s` hiển thị ngày cuối cùng mà hệ thống được bật nguồn (như trong `2019-08-05 20:13:22`). Kết quả của lệnh `uptime -s | sed -e 's/(.*) (.*)/\1/'` sẽ là gì?

4. Tùy chọn nào nên được cung cấp cho `grep` để nó đếm các dòng phù hợp thay vì hiển thị chúng?

Bài tập Mở rộng

1. Cấu trúc cơ bản của tệp HTML được bắt đầu bằng các phần tử `html`, `head` và `body`. Ví dụ:

   ```
   <html>
   <head>
    <title>News Site</title>
   </head>
   <body>
    <h1>Headline</h1>
    <p>Information of interest.</p>
   </body>
   </html>
   ```

   Hãy mô tả cách các địa chỉ có thể được sử dụng trong `sed` để chỉ hiển thị phần tử `body` và nội dung của nó.

2. Biểu thức `sed` nào sẽ xóa tất cả các thẻ khỏi tài liệu HTML và chỉ giữ lại văn bản được hiển thị?

3. Các tệp có phần mở rộng `.ovpn` là rất phổ biến trong việc định cấu hình máy khách VPN vì chúng không chỉ chứa cài đặt mà còn chứa cả nội dung của khóa và chứng nhận cho máy khách. Các khóa và chứng nhận này ban đầu nằm trong các tệp riêng biệt. Vì vậy, chúng cần được sao chép vào tệp `.ovpn`. Giả sử ta có đoạn trích sau đây của một mẫu `.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>
   ```

Giả sử các tệp `ca.crt`, `client.crt`, `client.key` và `ta.key` đều nằm trong thư mục hiện tại thì `sed` sẽ sửa đổi cấu hình mẫu như thế nào để thay thế từng tên tệp bằng nội dung của nó?

Tóm tắt

Bài học này nói về hai lệnh Linux quan trọng nhất liên quan đến biểu thức chính quy là `grep` và `sed`. Các tệp lệnh và lệnh phức hợp sẽ dựa vào `grep` và `sed` để thực thi nhiều tác vụ lọc và phân tích văn bản. Bài học đã đi qua các bước sau:

- Cách sử dụng `grep` và các biến thể của nó như `egrep` và `fgrep`.
- Cách sử dụng `sed` và hướng dẫn bên trong nó để thao tác với văn bản.
- Ví dụ về các ứng dụng của biểu thức chính quy sử dụng `grep` và `sed`.

Đáp án Bài tập Hướng dẫn

1. Lệnh `last` sẽ hiển thị danh sách người dùng đăng nhập lần cuối, bao gồm cả IP gốc của họ. Lệnh `egrep` sẽ được sử dụng như thế nào để lọc đầu ra `last` sao cho chỉ hiển thị các lần xuất hiện của địa chỉ IPv4 và loại bỏ mọi thông tin bổ sung trong dòng tương ứng?

   ```
   last -i | egrep -o '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}'
   ```

2. Tùy chọn nào nên được cung cấp cho `grep` để lọc chính xác đầu ra được tạo bởi lệnh `find` được thực thi với tùy chọn `-print0`?

   Tùy chọn `-z` hoặc `--null-data` như trong `find . -print0 | grep -z expression`.

3. Lệnh `uptime -s` hiển thị ngày cuối cùng mà hệ thống được bật nguồn (như trong `2019-08-05 20:13:22`). Kết quả của lệnh `uptime -s | sed -e 's/(.*) (.*)/\1/'` sẽ là gì?

   Một lỗi sẽ xảy ra. Theo mặc định, dấu ngoặc đơn phải được thoát để sử dụng tham chiếu ngược trong `sed`.

4. Tùy chọn nào nên được cung cấp cho `grep` để nó đếm các dòng phù hợp thay vì hiển thị chúng?

   Tuỳ chọn `-c`.

Đáp án Bài tập Mở rộng

1. Cấu trúc cơ bản của tệp HTML được bắt đầu bằng các phần tử `html`, `head` và `body`. Ví dụ:

```
<html>
<head>
 <title>News Site</title>
</head>
<body>
 <h1>Headline</h1>
 <p>Information of interest.</p>
</body>
</html>
```

   Hãy mô tả cách các địa chỉ có thể được sử dụng trong `sed` để chỉ hiển thị phần tử `body` và nội dung của nó.

   Để chỉ hiển thị `body`, các địa chỉ phải là `/<body>/,/<\/body>/` như trong `sed -n -e '/<body>/,/<\/body>/p '`. Tùy chọn `-n` được cung cấp cho `sed` để nó không in các dòng theo mặc định, vì thế mà ta nên sử dụng lệnh `p` ở cuối biểu thức `sed` để in các dòng phù hợp.

2. Biểu thức `sed` nào sẽ xóa tất cả các thẻ khỏi tài liệu HTML và chỉ giữ lại văn bản được hiển thị?

   Biểu thức `sed s/<[^>]*>//g` sẽ thay thế bất kỳ nội dung nào bên trong `<>` bằng một chuỗi rỗng.

3. Các tệp có phần mở rộng `.ovpn` là rất phổ biến trong việc định cấu hình máy khách VPN vì chúng không chỉ chứa cài đặt mà còn chứa cả nội dung của khóa và chứng nhận cho máy khách. Các khóa và chứng nhận này ban đầu nằm trong các tệp riêng biệt. Vì vậy, chúng cần được sao chép vào tệp `.ovpn`. Giả sử ta có đoạn trích sau đây của một mẫu `.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>
```

Giả sử các tệp `ca.crt`, `client.crt`, `client.key` và `ta.key` đều nằm trong thư mục hiện tại thì `sed` sẽ sửa đổi cấu hình mẫu như thế nào để thay thế từng tên tệp bằng nội dung của nó?

Lệnh

```
sed -r -e 's/(^[^.]*)\.(crt|key)$/cat \1.\2/e' < client.template > client.ovpn
```

sẽ thay thế bất kỳ dòng nào kết thúc bằng `.crt` hoặc `.key` với nội dung của tệp có tên trùng với dòng đó. Tùy chọn `-r` sẽ yêu cầu `sed` sử dụng các biểu thức chính quy mở rộng, trong khi `e` ở cuối biểu thức sẽ yêu cầu `sed` thay thế các kết quả trùng khớp với đầu ra của lệnh `cat \1.\2`. Các tham chiếu ngược `\1` và `\2` sẽ tương ứng với tên tệp và phần mở rộng được tìm thấy trong kết quả trùng khớp.

103.8 Chỉnh sửa Tệp cơ bản

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 103.8

Khối lượng

3

Các lĩnh vực kiến thức chính

- Điều hướng tài liệu bằng vi.
- Hiểu và sử dụng chế độ vi.
- Chèn, chỉnh sửa, xóa, sao chép và tìm văn bản trong vi.
- Kiến thức về Emacs, nano và vim.
- Cấu hình trình soạn thảo tiêu chuẩn.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

- `vi`
- `/`, `?`
- `h`, `j`, `k`, `l`
- `i`, `o`, `a`
- `d`, `p`, `y`, `dd`, `yy`
- `ZZ`, `:w!`, `:q!`
- `EDITOR`

103.8 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	103 Lệnh GNU và Unix
Mục tiêu:	103.8 Chỉnh sửa Tệp cơ bản
Bài:	1 trên 1

Giới thiệu

Trong hầu hết các bản phân phối Linux, `vi` — viết tắt của "visual" — luôn được cài đặt sẵn. Đây là trình soạn thảo tiêu chuẩn trong môi trường vỏ. Vi là một trình soạn thảo văn bản tương tác, nó sẽ hiển thị nội dung của tệp lên màn hình khi đang soạn thảo. Từ đó, nó cho phép người dùng di chuyển qua lại và thực hiện các sửa đổi ở bất kỳ đâu bên trong tài liệu. Tuy nhiên, không giống như các trình chỉnh sửa trực quan từ màn hình nền đồ họa, trình chỉnh sửa `vi` là một ứng dụng vỏ với các phím tắt cho mọi tác vụ chỉnh sửa. Có một chương trình thay thế cho `vi` được gọi là `vim` (*vi cải tiến*) đôi khi được sử dụng như một sự thay thế hiện đại cho `vi`. Bên cạnh nhiều cải tiến, `vim` còn hỗ trợ đánh dấu nổi cú pháp, hoàn tác/làm lại đa cấp độ và chỉnh sửa nhiều tài liệu. Mặc dù có nhiều tính năng hơn nhưng `vim` vẫn hoàn toàn tương thích ngược với `vi` và ta gần như sẽ không thể phân biệt được chúng trong hầu hết các tác vụ.

Cách tiêu chuẩn để bắt đầu `vi` là cung cấp cho nó đường dẫn đến tệp dưới dạng tham số. Để chuyển trực tiếp đến một dòng cụ thể, số của nó phải được thông báo bằng dấu cộng (như trong `vi +9 /etc/fstab` để mở `/etc/fstab/` và đặt con trỏ ở dòng thứ 9; nếu không có số, dấu cộng sẽ tự đặt con trỏ ở dòng cuối cùng).

Giao diện của `vi` rất đơn giản: toàn bộ không gian có sẵn trong cửa sổ dòng lệnh sẽ được sử dụng

để hiển thị tệp (thường được thông báo dưới dạng đối số lệnh) cho người dùng. Dấu hiệu trực quan duy nhất là một dòng chân trang hiển thị vị trí hiện tại của con trỏ và dấu ngã `~` cho biết nơi tệp kết thúc. Có các chế độ thực thi khác nhau cho `vi` khi hành vi của chương trình thay đổi, phổ biến nhất là *chế độ chèn* và *chế độ thường*.

Chế độ Chèn

Chế độ chèn rất đơn giản: văn bản sẽ xuất hiện trên màn hình khi nó được gõ trên bàn phím. Đây là kiểu tương tác mà đa số người dùng đều mong đợi từ một trình soạn thảo văn bản, nhưng nó không phải là phương thức trình bày tài liệu đầu tiên của `vi`. Để vào chế độ chèn, người dùng phải thực hiện lệnh chèn ở chế độ thường. Phím `Esc` sẽ kết thúc chế độ chèn và trở về chế độ thường - tức chế độ `vi` mặc định.

NOTE

Nếu muốn biết thêm về các chế độ thực thi khác, hãy mở `vi` và gõ:

```
:help vim-modes-intro
```

Chế độ Thường

Chế độ thường — còn được gọi là chế độ lệnh — là cách `vi` bắt đầu theo mặc định. Ở chế độ này, các phím trên bàn phím được liên kết với các lệnh cho các tác vụ điều hướng và thao tác văn bản. Hầu hết lệnh trong chế độ này là các phím đơn. Một số phím và chức năng của chúng ở chế độ thường là:

`0`, **`$`**
: Đi tới đầu và cuối dòng.

`1G`, **`G`**
: Đi tới đầu và cuối tài liệu.

`(`, **`)`**
: Đi tới đầu và cuối câu.

`{`, **`}`**
: Đi tới đầu và cuối đoạn văn.

`w`, **`W`**
: ??????Nhảy tới từ và nhảy tới từ bao gồm cả dấu chấm câu.

h, **j**, **k**, **l**

Trái, xuống, lên, phải.

e hoặc **E**

Đi tới cuối từ hiện tại.

/, **?**

Tìm kiếm tiến và lùi.

i, **I**

Chèn vào trước vị trí con trỏ hiện tại và ở đầu dòng hiện tại.

a, **A**

Chèn vào sau vị trí con trỏ hiện tại và ở cuối dòng hiện tại.

o, **o**

Thêm một dòng mới và chèn vào dòng tiếp theo hoặc dòng trước.

s, **S**

Xóa ký tự dưới con trỏ hoặc toàn bộ dòng và vào chế độ chèn.

c

Thay đổi (các) ký tự bên dưới con trỏ.

r

Thay thế ký tự dưới con trỏ.

x

Xóa các ký tự đã chọn hoặc ký tự bên dưới con trỏ.

v, **v**

Bắt đầu một vùng chọn mới với ký tự hiện tại hoặc toàn bộ dòng.

y, **yy**

Sao chép (kéo) các ký tự hoặc toàn bộ dòng.

p, **p**

Dán nội dung đã sao chép vào trước hoặc sau vị trí hiện tại.

u

Hoàn tác hành động cuối cùng.

`Ctrl-R`

Làm lại hành động cuối cùng.

`ZZ`

Đóng và lưu.

`ZQ`

Đóng và không lưu.

Nếu có một số đứng trước, lệnh sẽ được thực thi với đúng số lần đó. Ví dụ: nhấn `3yy` để sao chép dòng hiện tại cộng với hai dòng tiếp theo, nhấn `d5w` để xóa từ hiện tại và 4 từ tiếp theo, v.v.

Hầu hết các tác vụ chỉnh sửa đều là sự kết hợp giữa các lệnh. Ví dụ: chuỗi phím `vey` được sử dụng để sao chép một vùng chọn bắt đầu từ vị trí hiện tại cho đến hết từ hiện tại. Việc lặp lại lệnh cũng có thể được sử dụng trong các kết hợp. Vì vậy, `v3ey` sẽ sao chép một vùng chọn bắt đầu từ vị trí hiện tại cho đến khi kết thúc từ thứ ba tính từ đó.

`vi` có thể sắp xếp văn bản đã sao chép trong các thanh ghi và cho phép giữ các nội dung riêng biệt cùng một lúc. Một thanh ghi sẽ được chỉ định bởi một ký tự với ký tự `"` đứng trước. Sau khi được tạo, nó sẽ được lưu giữ cho đến khi kết thúc phiên hiện tại. Chuỗi phím `"ly` sẽ tạo một thanh ghi chứa vùng chọn hiện tại có thể được truy cập thông qua phím `l` . Sau đó, thanh ghi `l` có thể được dán bằng `"lp`.

Ngoài ra, chúng ta còn có một cách để đặt các dấu tùy chỉnh ở các vị trí tùy ý dọc theo văn bản để giúp ta dễ dàng và nhanh chóng di chuyển qua lại giữa chúng. Các dấu này được tạo bằng cách nhấn phím `m` và sau đó nhấn một phím để xác định vị trí hiện tại. Sau khi thực hiện xong, con trỏ sẽ quay lại vị trí đã đánh dấu khi nhấn `'` và theo sau là phím đã chọn.

Bất kỳ một chuỗi phím nào cũng có thể được ghi lại dưới dạng một macro để thực hiện trong tương lai. Ví dụ, một macro có thể được ghi lại để bao quanh một đoạn văn bản đã chọn trong dấu trích dẫn kép. Đầu tiên, một chuỗi văn bản sẽ được chọn và phím `q` được nhấn, tiếp theo là một phím thanh ghi để liên kết với macro, chẳng hạn như `d`. Dòng `recording @d` sẽ xuất hiện ở dòng chân trang, cho biết rằng quá trình ghi đang hoạt động. Vì giả định rằng một đoạn văn bản đã được chọn nên lệnh đầu tiên là `x` sẽ xóa (và tự động sao chép) văn bản đã chọn. Sau đó, ta nhấn phím `i` để chèn hai dấu trích dẫn kép vào vị trí hiện tại, sau đó là `Esc` để trở về chế độ thường. Lệnh cuối cùng là `P` dùng để chèn lại vùng chọn đã xóa ngay trước dấu trích dẫn kép cuối cùng. Bây giờ, việc nhấn lại `q` sẽ kết thúc quá trình ghi. Một macro bao gồm chuỗi `x`, `i`, `""`, `Esc` và `P` sẽ thực thi mỗi khi các phím `@d` được nhấn ở chế độ thường, trong đó, `d` là phím thanh ghi được liên kết với macro.

Tuy nhiên, macro sẽ chỉ khả dụng trong phiên hiện tại. Để làm cho macro tồn tại lâu dài, chúng

phải được lưu trữ trong tệp cấu hình. Vì hầu hết các bản phân phối hiện đại đều sử dụng *vim* làm trình chỉnh sửa tương thích với *vi* nên tệp cấu hình của người dùng sẽ là `~/.vimrc`. Bên trong `~/.vimrc`, dòng `let @d = 'xi""^[P'` sẽ đặt thanh ghi `d` thành chuỗi phím bên trong dấu trích dẫn đơn. Ta có thể sử dụng cùng một thanh ghi đã gán trước đó cho một macro để dán chuỗi phím của nó.

Lệnh Dấu hai chấm

Chế độ thường cũng hỗ trợ một tập hợp các lệnh `vi` khác là các lệnh *dấu hai chấm*. Đúng như tên của nó, các lệnh dấu hai chấm được thực thi sau khi ta nhấn phím dấu hai chấm `:` ở chế độ thường. Các lệnh dấu hai chấm cho phép người dùng thực hiện tìm kiếm, lưu, thoát, chạy các lệnh vỏ, thay đổi cài đặt `vi`, v.v. Để quay lại chế độ thường, ta phải thực thi lệnh `:visual` hoặc nhấn phím Enter không đi kèm với bất kỳ lệnh nào. Một số lệnh dấu hai chấm phổ biến nhất sẽ được chỉ ra sau đây (chữ cái đầu tiên không phải là một phần của lệnh):

`:s/REGEX/TEXT/g`

Thay thế tất cả các lần xuất hiện của biểu thức chính quy `REGEX` bằng `TEXT` trong dòng hiện tại. Nó chấp nhận cùng một cú pháp của lệnh `sed` bao gồm cả địa chỉ.

`:!`

Chạy lệnh vỏ sau.

`:quito` hoặc `:q`

Thoát khỏi chương trình.

`:quit!` hoặc `:q!`

Thoát khỏi chương trình mà không lưu.

`:wq`

Lưu và thoát.

`:exit` hoặc `:x` hoặc `:e`

Lưu và thoát nếu cần.

`:visual`

Quay trở lại chế độ điều hướng.

Chương trình `vi` tiêu chuẩn có khả năng thực hiện hầu hết các tác vụ chỉnh sửa văn bản, nhưng bất kỳ trình soạn thảo phi đồ họa nào khác cũng đều có thể được sử dụng để chỉnh sửa tệp văn bản trong môi trường vỏ.

TIP

Người dùng mới có thể sẽ gặp khó khăn trong việc ghi nhớ tất cả các phím lệnh của `vi` cùng một lúc. Các bản phân phối sử dụng `vim` cũng có lệnh `vimtutor` sử dụng `vim` để mở hướng dẫn từng bước về các hoạt động chính. Tệp là một bản sao có thể chỉnh sửa và được sử dụng để thực hành các lệnh cũng như dần dần làm quen với chúng.

Trình chỉnh sửa thay thế

Người dùng không quen với `vi` có thể sẽ gặp khó khăn khi thích ứng với nó vì hoạt động của nó không trực quan. Một giải pháp thay thế đơn giản hơn là GNU `nano` - một trình soạn thảo văn bản nhỏ cung cấp tất cả các tính năng chỉnh sửa văn bản cơ bản như hoàn tác/làm lại, đánh dấu nổi cú pháp, tìm kiếm và thay thế tương tác, tự động thụt lề, đánh số dòng, hoàn thành từ, khóa tệp, sao lưu tệp và hỗ trợ quốc tế hóa. Không giống như `vi`, tất cả các lần nhấn phím sẽ chỉ được chèn vào tài liệu đang được chỉnh sửa. Các lệnh trong `nano` được cung cấp bằng cách sử dụng phím `Ctrl` hoặc phím Meta (tùy thuộc vào loại hệ thống, Meta có thể là `Alt` hoặc `▯`).

`Ctrl-6` hoặc `Meta-A`

Bắt đầu một vùng chọn mới. Cũng có thể tạo vùng chọn bằng cách nhấn Shift và di chuyển con trỏ.

`Meta-6`

Sao chép vùng chọn hiện tại.

`Ctrl-K`

Cắt vùng chọn hiện tại.

`Ctrl-U`

Dán nội dung đã sao chép.

`Meta-U`

Hoàn tác.

`Meta-E`

Làm lại.

`Ctrl-\`

Thay thế văn bản tại vùng chọn.

`Ctrl-T`

Bắt đầu phiên kiểm tra chính tả cho tài liệu hoặc vùng chọn hiện tại.

Emacs là một trình soạn thảo văn bản rất phổ biến khác cho môi trường vỏ. Trong khi văn bản có

thể được đưa vào chỉ bằng cách nhập qua bàn phím (như trong `nano`) thì việc điều hướng trong tài liệu lại được hỗ trợ bởi các lệnh trên bàn phím (như trong `vi`). Emacs bao gồm nhiều tính năng giúp nó không chỉ đơn giản là một trình soạn thảo văn bản mà còn là một IDE (*môi trường phát triển tích hợp*) có khả năng biên dịch, chạy và thử nghiệm các chương trình. Emacs có thể được cấu hình như một ứng dụng khách email, tin tức hoặc RSS và khiến nó trở thành một bộ ứng dụng có năng suất rất cao.

Bản thân vỏ sẽ chạy một trình soạn thảo văn bản mặc định (thường là `vi`) mỗi khi cần thiết. Điều này cũng đúng trong trường hợp ví dụ như khi `crontab -e` được thực thi để chỉnh sửa *các công việc định kỳ* (cronjobs). Bash sử dụng các biến phiên `VISUAL` hoặc `EDITOR` để tìm ra trình soạn thảo văn bản mặc định cho môi trường vỏ. Ví dụ: lệnh `export EDITOR=nano` sẽ xác định `nano` là trình soạn thảo văn bản mặc định trong phiên vỏ hiện tại. Để thực hiện thay đổi này liên tục trong các phiên, lệnh phải được đưa vào `~/.bash_profile`.

Bài tập Hướng dẫn

1. `vi` được sử dụng nhiều nhất cho việc chỉnh sửa các tệp cấu hình và mã nguồn nơi việc thụt lề sẽ giúp xác định các phần của văn bản. Một vùng chọn có thể được thụt vào bên trái bằng cách nhấn `<` và bên phải bằng cách nhấn `>`. Những phím nào nên được nhấn ở chế độ thường để thụt lề ba bước sang trái cho vùng chọn hiện tại?

2. Ta có thể chọn toàn bộ dòng bằng cách nhấn `V` ở chế độ thường của `vi`. Tuy nhiên, ký tự xuống dòng kết thúc cũng sẽ được bao gồm. Những phím nào nên được nhấn ở chế độ thường để chọn từ ký tự bắt đầu cho đến cuối dòng nhưng không bao gồm ký tự xuống dòng?

3. `vi` nên được thực thi như thế nào trong dòng lệnh để mở `~/.bash_profile` và chuyển thẳng đến dòng cuối cùng?

4. Những phím nào nên được nhấn trong chế độ thường của `vi` để xóa các ký tự từ vị trí con trỏ hiện tại cho đến ký tự dấu chấm tiếp theo?

Bài tập Mở rộng

1. `vim` cho phép chọn các khối văn bản có chiều rộng tùy ý chứ không chỉ các phần có toàn bộ dòng. Bằng cách nhấn `Ctrl` + `V` ở chế độ thường, một vùng chọn sẽ được thực hiện bằng cách di chuyển con trỏ lên, xuống, sang trái và phải. Bằng cách sử dụng phương pháp này, làm cách nào để xóa một khối bắt đầu từ ký tự đầu tiên trong dòng hiện tại có chứa tám cột và năm dòng văn bản tiếp theo?

2. Phiên `vi` bị gián đoạn do mất điện đột xuất. Khi mở lại tệp, `vi` sẽ nhắc người dùng nếu họ muốn khôi phục tệp hoán đổi (bản sao tự động do `vi` tạo). Người dùng nên làm gì để loại bỏ tệp hoán đổi?

3. Trong một phiên `vim`, trước đó đã có một dòng được sao chép vào thanh ghi `l`. Tổ hợp phím nào sẽ ghi một macro trong thanh ghi `a` để dán dòng trong thanh ghi `l` vào ngay trước dòng hiện tại?

Tóm tắt

Bài học này đã nói về trình soạn thảo văn bản tiêu chuẩn cho môi trường vỏ Linux là `vi`. Mặc dù có vẻ hơi phức tạp đối với những người dùng mới nhưng `vi` có các tính năng khiến nó trở thành một lựa chọn tốt trong việc chỉnh sửa văn bản kỹ thuật và phi kỹ thuật. Bài học đã đi qua các bước sau:

- Cách sử dụng cơ bản và các tính năng hữu ích của `vi`.
- `vim` — tức `vi` cải tiến — là gì và các trình soạn thảo thay thế khác.
- Cách xác định trình soạn thảo văn bản mặc định cho môi trường vỏ.

Các lệnh và tiến trình đã được nhắc đến là:

- Trình chỉnh sửa `vi` và phiên bản cải tiến của nó `vim`.
- Chỉnh sửa văn bản cơ bản trong `vi`.
- Trình chỉnh sửa thay thế `emacs` và `nano`.

Đáp án Bài tập Hướng dẫn

1. `vi` được sử dụng nhiều nhất cho việc chỉnh sửa các tệp cấu hình và mã nguồn nơi việc thụt lề sẽ giúp xác định các phần của văn bản. Một vùng chọn có thể được thụt vào bên trái bằng cách nhấn `<` và bên phải bằng cách nhấn `>`. Những phím nào nên được nhấn ở chế độ thường để thụt lề ba bước sang trái cho vùng chọn hiện tại?

   `3<`, tức ba bước về bên trái.

2. Ta có thể chọn toàn bộ dòng bằng cách nhấn `V` ở chế độ thường của `vi`. Tuy nhiên, ký tự xuống dòng kết thúc cũng sẽ được bao gồm. Những phím nào nên được nhấn ở chế độ thường để chọn từ ký tự bắt đầu cho đến cuối dòng nhưng không bao gồm ký tự xuống dòng?

   Các phím `0v$h`, tức `0` ("nhảy đến đầu dòng"), `v` ("bắt đầu chọn ký tự"), `$` ("đi đến cuối dòng`) ") và `h` ("`quay lại một vị trí").

3. `vi` nên được thực thi như thế nào trong dòng lệnh để mở `~/.bash_profile` và chuyển thẳng đến dòng cuối cùng?

   Lệnh `vi + ~/.bash_profile` sẽ mở tệp và đặt con trỏ ở dòng cuối cùng.

4. Những phím nào nên được nhấn trong chế độ thường của `vi` để xóa các ký tự từ vị trí con trỏ hiện tại cho đến ký tự dấu chấm tiếp theo?

   Các phím `dt.`, tức `d` ("bắt đầu xóa"), `t` ("nhảy tới ký tự sau") và `.` (ký tự dấu chấm).

Đáp án Bài tập Mở rộng

1. `vim` cho phép chọn các khối văn bản có chiều rộng tùy ý chứ không chỉ các phần có toàn bộ dòng. Bằng cách nhấn `Ctrl` + `V` ở chế độ thường, một vùng chọn sẽ được thực hiện bằng cách di chuyển con trỏ lên, xuống, sang trái và phải. Bằng cách sử dụng phương pháp này, làm cách nào để xóa một khối bắt đầu từ ký tự đầu tiên trong dòng hiện tại có chứa tám cột và năm dòng văn bản tiếp theo?

   Tổ hợp `0`, `Ctrl-V` và `8l5jd` sẽ chọn và xóa khối tương ứng.

2. Phiên `vi` bị gián đoạn do mất điện đột xuất. Khi mở lại tệp, `vi` sẽ nhắc người dùng nếu họ muốn khôi phục tệp hoán đổi (bản sao tự động do `vi` tạo). Người dùng nên làm gì để loại bỏ tệp hoán đổi?

   Nhấn `d` khi được nhắc bởi `vi`.

3. Trong một phiên `vim`, trước đó đã có một dòng được sao chép vào thanh ghi `l`. Tổ hợp phím nào sẽ ghi một macro trong thanh ghi `a` để dán dòng trong thanh ghi `l` vào ngay trước dòng hiện tại?

   Tổ hợp `qa"lPq`, tức `q` (“bắt đầu ghi macro”), `a` (“gán thanh ghi `a` cho macro”), `"l` (“chọn văn bản trong thanh ghi `l`”), `P` (“dán trước dòng hiện tại”) và `q` (“kết thúc ghi macro”).

Chủ đề 104: Thiết bị, Hệ thống tệp Linux, Tiêu chuẩn phân cấp hệ thống Tệp

104.1 Tạo Phân vùng và Hệ thống Tệp

Tham khảo các mục tiêu LPI

LPIC-1 v5, Exam 101, Objective 104.1

Khối lượng

2

Các lĩnh vực kiến thức chính

- Quản lý bảng phân vùng MBR và GPT
- Sử dụng các lệnh mkfs khác nhau để tạo các tệp hệ thống khác nhau, chẳng hạn như:
  - ext2/ext3/ext4
  - XFS
  - VFAT
  - exFAT
- Kiến thức về tính năng cơ bản của Btrfs (bao gồm các hệ thống tệp đa thiết bị, nén và các ổ phân vùng con).

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

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

104.1 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	104 Thiết bị, Hệ thống tệp Linux, Tiêu chuẩn phân cấp hệ thống Tệp
Mục tiêu:	104.1 Tạo Phân vùng và Hệ thống Tệp
Bài:	1 trên 1

Giới thiệu

Trên bất kỳ hệ điều hành nào, mỗi một đĩa cứng đều cần được phân vùng trước khi có thể sử dụng. Phân vùng là một tập hợp con logic của đĩa vật lý và thông tin về các phân vùng sẽ được lưu trữ trong bảng phân vùng. Bảng này bao gồm thông tin về các cung đầu tiên và cuối cùng của phân vùng, loại của nó cũng như các chi tiết khác về từng phân vùng.

Thông thường, mỗi một phân vùng đều được hệ điều hành coi là một "ổ" riêng biệt ngay cả khi chúng đều nằm trong cùng một phương tiện vật lý. Trên các hệ thống Windows, chúng được gán các chữ cái như `C:` (từ trước tới nay vẫn được coi là ổ chính), `D:`, v.v. Trên Linux, mỗi phân vùng sẽ được gán cho một thư mục trong `/dev`, như `/dev/sda1` hoặc `/dev/sda2`.

Trong bài học này, chúng ta sẽ học cách tạo, xóa, khôi phục và thay đổi kích thước phân vùng bằng ba tiện ích phổ biến nhất (`fdisk`, `gdisk` và `parted`), cách tạo hệ thống tệp trên chúng cũng như cách tạo và thiết lập một *phân vùng hoán đổi* hoặc *tệp hoán đổi* để dùng làm bộ nhớ ảo.

NOTE Vì các lý do về tiền lệ, trong bài học này, chúng ta sẽ gọi phương tiện lưu trữ là "đĩa" dù các hệ thống lưu trữ hiện đại như SSD và Bộ lưu trữ Flash hoàn toàn

không hề chứa bất kỳ một "đĩa" nào.

Hiểu về MBR và GPT

Có hai cách chính để lưu trữ thông tin phân vùng trên ổ cứng. Cách đầu tiên là MBR (*Bản ghi Khởi động Chính*) và cách thứ hai là GPT (*Bảng phân vùng GUID*).

MBR

Đây là tàn dư từ những ngày đầu của MS-DOS (cụ thể hơn là PC-DOS 2.0 từ năm 1983) và trong nhiều thập kỷ được coi là lược đồ phân vùng tiêu chuẩn trên PC. Bảng phân vùng được lưu trữ trên cung đầu tiên của đĩa được gọi là *Cung Khởi động*, cùng với đó còn có trình tải khởi động (trên các hệ thống Linux thường là trình tải khởi động *GRUB*). Nhưng MBR có một loạt hạn chế gây cản trở việc sử dụng nó trên các hệ thống hiện đại như việc không thể xử lý các đĩa có kích thước lớn hơn 2TB và giới hạn chỉ có 4 phân vùng chính trên mỗi đĩa.

GUID

Một hệ thống phân vùng giải quyết nhiều hạn chế của MBR. Nó không có giới hạn thực tế về kích thước đĩa và số lượng phân vùng tối đa chỉ bị giới hạn bởi chính hệ điều hành. Nó thường được tìm thấy trên các máy hiện đại hơn sử dụng UEFI thay vì BIOS PC cũ.

Trong các tác vụ quản trị hệ thống, người dùng sẽ có thể tìm thấy cả hai lược đồ này đang được sử dụng. Vì vậy, chúng ta phải biết cách sử dụng các công cụ liên quan đến từng lược đồ để tạo, xóa hoặc sửa đổi các phân vùng.

Quản lý phân vùng MBR với FDISK

Tiện ích tiêu chuẩn để quản lý phân vùng MBR trên Linux là `fdisk`. Đây là một tiện ích tương tác được điều khiển bằng menu. Để sử dụng nó, hãy nhập `fdisk`, theo sau là tên thiết bị tương ứng với đĩa cần chỉnh sửa. Ví dụ như lệnh

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

sẽ chỉnh sửa bảng phân vùng của thiết bị được kết nối SATA đầu tiên (`sda`) trên hệ thống. Hãy nhớ rằng chúng ta cần chỉ định thiết bị tương ứng với đĩa vật lý chứ không phải một trong các phân vùng của nó (như `/dev/sda1`).

NOTE Tất cả các hoạt động liên quan đến đĩa trong bài học này cần phải được thực hiện với tư cách là siêu người dùng `root` (quản trị viên hệ thống) hoặc với quyền gốc bằng cách sử dụng `sudo`.

Khi được gọi, `fdisk` sẽ hiển thị lời chào, sau đó là một lời cảnh báo và nó sẽ chờ lệnh của người dùng.

```
# 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):
```

Cảnh báo là một yếu tố quan trọng. Chúng ta có thể tạo, chỉnh sửa hoặc xóa phân vùng theo ý muốn nhưng sẽ *không có gì* được ghi vào đĩa trừ khi ta sử dụng lệnh ghi (`w`). Vì vậy, ta có thể "thực hành" mà không sợ nguy cơ mất dữ liệu, miễn là ta tránh xa phím `w`. Để thoát `fdisk` mà không lưu các thay đổi, hãy sử dụng lệnh `q`.

NOTE Dù vậy chúng ta không bao giờ nên thực hành trên một đĩa quan trọng vì dù sao vẫn luôn có tồn tại nhiều rủi ro. Thay vào đó, hãy sử dụng ổ đĩa ngoại vi dự phòng hoặc ổ USB flash.

In Bảng Phân vùng hiện tại

Lệnh `p` được sử dụng để in bảng phân vùng hiện tại. Đầu ra sẽ trông giống như sau:

```
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
```

Dưới đây là ý nghĩa của từng cột:

`Device`

Thiết bị được gán cho phân vùng.

`Boot`

Cho biết phân vùng “có thể khởi động” được hay không.

`Start`

Cung bắt đầu phân vùng.

`End`

Cung kết thúc phân vùng.

`Sectors`

Tổng số cung trong phân vùng. Nhân nó với kích thước cung để có được kích thước phân vùng tính bằng byte.

`Size`

Kích thước của phân vùng ở định dạng “con người có thể đọc được”. Trong ví dụ trên, các giá trị được tính bằng gigabyte.

`Id`

Giá trị số đại diện cho loại phân vùng.

`Type`

Mô tả loại phân vùng.

Phân vùng Chính và Phân vùng Mở rộng

Trên đĩa MBR có thể có 2 loại phân vùng cơ bản là phân vùng *chính* và *mở rộng*. Như đã nói trước đây, ta chỉ có thể có 4 phân vùng chính trên đĩa và nếu muốn tạo đĩa “có thể khởi động được” thì phân vùng đầu tiên phải là một phân vùng chính.

Một cách để khắc phục hạn chế này là tạo một phân vùng mở rộng hoạt động như một vùng chứa cho các phân vùng *logic*. Ví dụ, chúng ta có thể có một phân vùng chính, một phân vùng mở rộng chiếm phần còn lại của dung lượng đĩa và năm phân vùng logic bên trong nó.

Đối với một hệ điều hành như Linux, các phân vùng chính và mở rộng đều được xử lý theo cùng một cách. Do đó, không có một “lợi thế” nào khi chúng ta sử dụng phân vùng này thay vì phân vùng còn lại.

Tạo một Phân vùng

Để tạo một phân vùng, chúng ta sẽ sử dụng lệnh `n`. Theo mặc định, các phân vùng sẽ được tạo khi dung lượng đĩa chưa được phân bổ. Chúng ta sẽ được hỏi về loại phân vùng (chính hoặc mở rộng),

cung đầu tiên và cung cuối cùng.

Cung đầu tiên thường có thể chấp nhận giá trị mặc định do `fdisk` đề xuất trừ khi chúng ta cần một phân vùng để bắt đầu tại một cung cụ thể. Thay vì chỉ định cung cuối cùng, ta có thể chỉ định kích thước và theo sau là các chữ cái `K`, `M`, `G`, `T` hoặc `P` (Kilo, Mega, Giga, Tera hoặc Peta). Vì vậy, nếu muốn tạo phân vùng 1 GB, ta có thể chỉ định `+1G` là `Last sector` (Cung cuối cùng) và `fdisk` sẽ xác định kích cỡ phân vùng tương ứng. Hãy xem ví dụ sau về việc tạo phân vùng chính:

```
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
```

Kiểm tra Dung lượng chưa phân bổ

Nếu không rõ có bao nhiêu dung lượng trống trên đĩa, chúng ta có thể sử dụng lệnh `F` để hiển thị dung lượng chưa phân bổ như sau:

```
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
```

Xóa Phân vùng

Để xóa một phân vùng, hãy sử dụng lệnh `d`. `fdisk` sẽ hỏi về số phân vùng ta muốn xóa *trừ khi* chỉ có *duy nhất một* phân vùng trên đĩa. Trong trường hợp này, phân vùng này sẽ được *chọn và xóa ngay lập tức*.

Hãy lưu ý rằng nếu chúng ta xóa một phân vùng mở rộng, tất cả các phân vùng logic ở bên trong nó cũng sẽ bị xóa.

Hãy chú ý Khoảng cách!

Hãy nhớ rằng khi tạo một phân vùng mới bằng `fdisk`, kích thước tối đa sẽ bị giới hạn bởi lượng dung lượng *liền kề* tối đa chưa phân bổ tối đa trên đĩa. Ví dụ chúng ta có bản đồ phân vùng sau:

```
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
```

Sau đó, chúng ta xóa phân vùng 2 và kiểm tra dung lượng trống:

```
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
```

Nếu cộng cả kích thước của dung lượng chưa phân bổ thì trên lý thuyết, chúng ta có sẵn 881 MB dung lượng trống. Nhưng hãy xem điều gì sẽ xảy ra khi chúng ta cố gắng tạo một phân vùng 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.
```

Điều này xảy ra vì dung lượng chưa phân bổ liền kề lớn nhất trên đĩa là khối 512 MB thuộc về phân vùng 2. Phân vùng mới của chúng ta không thể "vượt qua" phân vùng 3 để sử dụng số dung lượng chưa phân bổ ở sau nó.

Thay đổi loại Phân vùng

Đôi khi chúng ta sẽ cần thay đổi loại phân vùng, đặc biệt là khi xử lý các ổ đĩa cứng được sử dụng trên các hệ điều hành và nền tảng khác. Điều này có thể được thực hiện bằng lệnh `t`, theo sau là số phân vùng cần thay đổi.

Loại phân vùng phải được chỉ định bằng mã thập lục phân tương ứng của nó và chúng ta có thể xem danh sách tất cả các mã hợp lệ bằng cách sử dụng lệnh `l`.

Đừng nhầm lẫn loại phân vùng với hệ thống tệp được sử dụng trên đó. Mặc dù trước đây giữa chúng có một mối quan hệ nhất định nhưng hiện nay điều này không phải lúc nào cũng đúng (ví dụ như một phân vùng Linux có thể chứa bất kỳ hệ thống tệp gốc Linux nào như *ext4* hoặc *ReiserFS*).

TIP Phân vùng Linux là loại `83` (Linux). Các phân vùng hoán đổi là loại `82` (Hoán đổi Linux).

Quản lý phân vùng GUID với GDISK

Tiện ích `gdisk` cũng tương đương với `fdisk` trong việc xử lý các đĩa được phân vùng GPT. Trên thực tế, giao diện của nó được mô phỏng theo `fdisk` với dấu nhắc tương tác và các lệnh tương tự (hoặc rất giống).

In Bảng Phân vùng hiện tại

Lệnh `p` được sử dụng để in bảng phân vùng hiện tại. Đầu ra của lệnh sẽ giống như sau:

```
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
```

Chúng ta có thể nhận ra một vài điểm khác biệt ngay trong phần đầu:

- Mỗi đĩa có một Mã định danh đĩa (GUID) duy nhất. Đây là số thập lục phân 128 bit được chỉ định ngẫu nhiên khi bảng phân vùng được tạo. Vì có $3{,}4 \times 10^{38}$ giá trị khả thi cho con số này nên khả năng 2 đĩa ngẫu nhiên có cùng GUID là rất thấp. GUID có thể được sử dụng để xác định hệ thống tệp nào sẽ được gắn vào lúc khởi động (và ở đâu), đồng thời loại bỏ nhu cầu sử dụng đường dẫn thiết bị để làm điều này (như `/dev/sdb`).
- Hãy xem cụm từ `Partition table holds up to 128 entries` (Bảng phân vùng chứa tối đa 128 mục nhập). Chúng ta đích thực là có thể có tối đa 128 phân vùng trên đĩa GPT. Do đó, ta sẽ không cần đến phân vùng *chính* và *mở rộng*.
- Dung lượng trống được liệt kê ở dòng cuối cùng, vì thế mà không cần tới lệnh tương đương với lệnh F từ fdisk.

Tạo Phân vùng

Lệnh để tạo phân vùng là `n` (cũng giống như trong `fdisk`). Sự khác biệt chính là bên cạnh số phân vùng và cung đầu tiên và cuối cùng (hoặc kích thước), ta cũng có thể chỉ định loại phân vùng trong quá trình tạo. Phân vùng GPT hỗ trợ nhiều loại hơn MBR. Chúng ta có thể kiểm tra danh sách tất cả các loại được hỗ trợ bằng cách sử dụng lệnh `l`.

Xóa Phân vùng

Để xóa một phân vùng, hãy nhập `d` và số phân vùng. Không giống như `fdisk`, phân vùng đầu tiên sẽ không được chọn tự động nếu nó là phân vùng duy nhất ở trên đĩa.

Trên đĩa GPT, các phân vùng có thể được sắp xếp lại dễ dàng (hoặc là "phân loại") để tránh các khoảng trống trong chuỗi đánh số. Để làm điều này, ta chỉ cần sử dụng lệnh `s`. Ví dụ: hãy tưởng tượng một đĩa có bảng phân vùng như sau:

```
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
```

Nếu xóa phân vùng thứ hai, bảng sẽ trở thành:

```
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
```

Nếu ta sử dụng lệnh `s`, nó sẽ trở thành:

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

Hãy chú ý việc phân vùng thứ ba đã trở thành phân vùng thứ hai.

Khoảng cách?

Không giống như đĩa MBR, khi tạo phân vùng trên đĩa GPT, kích thước của phân vùng sẽ không bị giới hạn bởi lượng dung lượng *liền kề* chưa phân bổ tối đa. Ta có thể tận dụng đến tận bit cuối cùng của một cung tự do dù nó nằm ở đâu trên đĩa.

Tùy chọn Khôi phục

Đĩa GPT lưu trữ các bản sao dự phòng của tiêu đề GPT và bảng phân vùng, giúp việc khôi phục đĩa trong trường hợp dữ liệu này bị hỏng trở nên dễ dàng. `gdisk` cung cấp các tính năng để hỗ trợ các tác vụ khôi phục đó và ta có thể truy cập chúng bằng lệnh `r`.

Chúng ta có thể xây dựng lại bảng phân vùng hoặc tiêu đề GPT chính bị hỏng bằng `b` và `c` tương ứng hoặc sử dụng tiêu đề và bảng chính để xây dựng lại bản sao lưu bằng `d` và `e`. Ta cũng có thể chuyển đổi MBR thành GPT bằng `f` và làm ngược lại với `g`. Nếu nhập `?` trong menu khôi phục ta sẽ nhận được danh sách tất cả các lệnh khôi phục có sẵn cũng như mô tả về chức năng của chúng.

Tạo Hệ thống Tệp

Phân vùng đĩa chỉ là bước đầu tiên khi sử dụng đĩa. Sau đó, chúng ta sẽ cần định dạng phân vùng bằng hệ thống tệp trước khi sử dụng nó để lưu trữ dữ liệu.

Một hệ thống tệp sẽ kiểm soát cách dữ liệu được lưu trữ và truy cập trên đĩa. Linux hỗ trợ nhiều hệ thống tệp, một trong số chúng là thuần Linux (như họ ext - Hệ thống tệp mở rộng), số còn lại đến từ những hệ điều hành khác như FAT từ MS-DOS, NTFS từ Windows NT, HFS và HFS+ từ Mac OS, v.v.

Công cụ tiêu chuẩn được sử dụng để tạo hệ thống tệp trên Linux là `mkfs` với nhiều "tuỳ chọn" tùy theo hệ thống tệp mà nó cần hoạt động.

Tạo Hệ thống Tệp ext2/ext3/ext4

Hệ thống Tệp mở rộng (ext) là hệ thống tệp đầu tiên dành cho Linux và qua nhiều năm đã được

thay thế bằng các phiên bản mới có tên là ext2, ext3 và ext4. Hiện nay đây là hệ thống tệp mặc định cho nhiều bản phân phối Linux.

Các tiện ích `mkfs.ext2`, `mkfs.ext3` và `mkfs.ext4` được sử dụng để tạo các hệ thống tệp ext2, ext3 và ext4. Trên thực tế, tất cả các "tiện ích" này chỉ tồn tại dưới dạng các liên kết tượng trưng đến một tiện ích khác có tên là `mke2fs`. `mke2fs` thay đổi các giá trị mặc định của nó theo tên mà nó được gọi. Như vậy, tất cả chúng đều có hành vi và tham số dòng lệnh giống nhau.

Hình thức sử dụng đơn giản nhất là:

```
# mkfs.ext2 TARGET
```

Trong đó, `TARGET` là tên của phân vùng nơi hệ thống tệp sẽ được tạo. Ví dụ: để tạo hệ thống tệp ext3 trên `/dev/sdb1`, lệnh sẽ là:

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

Thay vì sử dụng lệnh tương ứng với hệ thống tệp muốn tạo, ta có thể truyền tham số `-t` cho `mke2fs`, theo sau là tên hệ thống tệp. Ví dụ: các lệnh sau là tương đương và sẽ tạo một hệ thống tệp ext4 trên `/dev/sdb1`.

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

Tham số Dòng lệnh

`mke2fs` hỗ trợ nhiều tùy chọn và tham số dòng lệnh. Dưới đây là một trong những tham số quan trọng nhất. Tất cả những tham số này cũng áp dụng cho `mkfs.ext2`, `mkfs.ext3` và `mkfs.ext4`:

`-b SIZE`

Đặt kích thước của các khối dữ liệu trong thiết bị thành `SIZE`, có thể là 1024, 2048 hoặc 4096 byte cho mỗi khối.

`-c`

Kiểm tra thiết bị đích để tìm các khối hỏng trước khi tạo hệ thống tệp. Ta có thể chạy kiểm tra kỹ lưỡng nhưng chậm hơn nhiều bằng cách truyền tham số này hai lần (như trong `mkfs.ext4 -c -c TARGET`). }

`-d DIRECTORY`

Sao chép nội dung của thư mục được chỉ định vào thư mục gốc của hệ thống tệp mới. Nó sẽ rất hữu ích nếu ta cần "điền trước" đĩa với một tập hợp tệp được xác định trước.

`-F`

Cẩn thận! Tùy chọn này sẽ *buộc* mke2fs tạo một hệ thống tệp ngay cả khi các tùy chọn khác được truyền cho nó hoặc khi mục tiêu mang tính nguy hiểm hoặc không có ý nghĩa. Nếu được chỉ định hai lần (như trong `-F -F`), nó thậm chí có thể được sử dụng để tạo hệ thống tệp trên thiết bị đã được gắn kết hoặc đang sử dụng. Đây là một việc rất *rất* tồi tệ.

`-L VOLUME_LABEL`

Sẽ đặt nhãn ổ phân vùng thành nhãn được chỉ định trong `VOLUME_LABEL`. Nhãn này có thể dài tối đa 16 ký tự.

`-n`

Đây là một tùy chọn rất hữu ích. Nó sẽ mô phỏng việc tạo hệ thống tệp và hiển thị những gì sẽ được thực hiện nếu được thực thi mà không có tùy chọn `n`. Hãy coi nó như một chế độ "thử nghiệm". Việc kểm tra mọi thứ trước khi thực sự thực hiện bất kỳ thay đổi nào đối với đĩa là một việc rất hữu ích.

`-q`

Chế độ im lặng. `mke2fs` sẽ chạy bình thường nhưng sẽ không tạo ra bất kỳ đầu ra nào trong cửa sổ dòng lệnh. Nó sẽ khá hữu ích khi ta chạy `mke2fs` từ tệp lệnh.

`-U ID`

Thao tác này sẽ đặt UUID (*Mã Định danh duy nhất toàn cầu*) của một phân vùng thành giá trị được chỉ định làm ID. UUID là các số 128 bit trong ký hiệu thập lục phân dùng để xác định duy nhất một phân vùng cho hệ điều hành. Số này được chỉ định là một chuỗi 32 chữ số ở định dạng 8-4-4-4-12, nghĩa là 8 chữ số, gạch nối, 4 chữ số, gạch nối, 4 chữ số, gạch nối, 4 chữ số, gạch nối, 12 chữ số (như `D249E380 -7719-45A1-813C-35186883987E`). Thay vì ID, ta cũng có thể chỉ định các tham số như `clear` để xóa UUID của hệ thống tệp, `random` để sử dụng UUID được tạo ngẫu nhiên hoặc `time` để tạo UUID dựa trên thời gian.

`-V`

Chế độ chi tiết: in nhiều thông tin hơn trong quá trình hoạt động so với bình thường. Nó sẽ hữu ích cho mục đích gỡ lỗi.

Tạo một Hệ thống Tệp XFS

XFS là một hệ thống tệp hiệu suất cao do Silicon Graphics phát triển lần đầu tiên vào năm 1993

cho hệ điều hành IRIX của hãng. Do các tính năng về hiệu suất và độ tin cậy nên nó thường được sử dụng cho các máy chủ và các môi trường khác yêu cầu băng thông hệ thống tệp cao (hoặc được bảo đảm).

Các công cụ quản lý hệ thống tệp XFS là một phần của gói `xfsprogs`. Gói này có thể sẽ cần được cài đặt thủ công vì nó không mặc định có trong một số bản phân phối Linux. Những hệ điều hành khác (như Red Hat Enterprise Linux 7) sử dụng XFS làm hệ thống tệp mặc định.

Các hệ thống tệp XFS được chia thành ít nhất 2 phần, *phần nhật ký* nơi lưu trữ nhật ký của tất cả các hoạt động của hệ thống tệp (thường được gọi là *Nhật ký*) và phần *dữ liệu*. Phần nhật ký có thể được đặt bên trong phần dữ liệu (hành vi mặc định) hoặc thậm chí trên một đĩa riêng hoàn toàn để có hiệu suất và độ tin cậy cao hơn.

Lệnh cơ bản nhất để tạo hệ thống tệp XFS là `mkfs.xfs TARGET`, trong đó, `TARGET` là phân vùng cần hệ thống tệp được tạo trong đó (ví dụ: `mkfs.xfs /dev/sda1`).

Như trong `mke2fs`, `mkfs.xfs` cũng hỗ trợ một số tùy chọn dòng lệnh. Dưới đây là một trong số những tuỳ chọn phổ biến nhất.

`-b size=VALUE`

Đặt kích thước khối trên hệ thống tệp (tính bằng byte) thành kích thước được chỉ định trong `VALUE`. Giá trị mặc định là 4096 byte (4 KiB), tối thiểu là 512 và tối đa là 65536 (64 KiB).

`-m crc=VALUE`

Các tham số bắt đầu bằng `-m` là các tùy chọn siêu dữ liệu. Tuỳ chọn này sẽ cho phép (nếu `VALUE` là `1`) hoặc vô hiệu hóa (nếu `VALUE` là `0`) việc sử dụng kiểm tra CRC32c để xác minh tính toàn vẹn của tất cả các siêu dữ liệu trên đĩa. Điều này cho phép ta phát hiện và khôi phục lỗi tốt hơn sau các sự cố liên quan đến phần cứng. Do đó mà tính năng này được kích hoạt theo mặc định. Tác động hiệu suất của công tác kiểm tra này luôn ở mức tối thiểu nên thông thường không có lý do gì để vô hiệu hóa nó.

`-m uuid=VALUE`

Đặt UUID phân vùng thành giá trị được chỉ định trong VALUE. Hãy nhớ rằng UUID là các số gồm 32 ký tự (128 bit) ở cơ sở thập lục phân và được chỉ định trong nhóm 8, 4, 4, 4 và 12 chữ số được phân tách bằng dấu gạch ngang (như `1E83E3A3-3AE9-4AAC-BF7E-29DFFECD36C0`).

`-f`

Buộc tạo hệ thống tệp trên thiết bị đích ngay cả khi có một hệ thống tệp được phát hiện trên thiết bị đó.

`-l logdev=DEVICE`

Tuỳ chọn này sẽ đặt phần nhật ký của hệ thống tệp trên thiết bị đã chỉ định thay vì bên trong phần dữ liệu.

`-l size=VALUE`

Tuỳ chọn này sẽ đặt kích thước của phần nhật ký thành kích thước được chỉ định trong `VALUE`. Kích thước có thể được chỉ định bằng byte và có thể sử dụng các hậu tố như `m` hoặc `g`. Ví dụ: `-l size=10m` sẽ giới hạn phần nhật ký ở 10 Megabyte.

`-q`

Chế độ im lặng. Trong chế độ này, `mkfs.xfs` sẽ không in các tham số của hệ thống tệp được tạo.

`-L LABEL`

Thiết lập nhãn hệ thống tệp, có thể dài tối đa 12 ký tự.

`-N`

Tương tự như tham số `-n` của `mke2fs`, tuỳ chọn này sẽ làm cho `mkfs.xfs` in tất cả các tham số để tạo hệ thống tệp mà không thực sự tạo nó.

Tạo Hệ thống Tệp FAT hoặc VFAT

Hệ thống tệp FAT có nguồn gốc từ MS-DOS và qua nhiều năm đã có được nhiều bản sửa đổi, đỉnh cao là định dạng FAT32 được phát hành vào năm 1996 với Windows 95 OSR2.

VFAT là phần mở rộng của định dạng FAT16 với hỗ trợ tên tệp dài (tối đa 255 ký tự). Cả hai hệ thống tệp đều được xử lý bởi cùng một tiện ích là `mkfs.fat`. `mkfs.vfat` chính là bí danh của nó.

Hệ thống tệp FAT có những nhược điểm quan trọng hạn chế việc sử dụng nó trên các ổ đĩa cứng lớn. Ví dụ: FAT16 hỗ trợ ổ phân vùng có dung lượng tối đa là 4 GB và kích thước tệp tối đa là 2 GB. FAT32 tăng kích thước ổ phân vùng lên tối đa 2 PB và kích thước tệp tối đa lên 4 GB. Do đó, các hệ thống tệp FAT ngày nay được sử dụng phổ biến hơn trên các ổ đĩa flash nhỏ hoặc thẻ nhớ (kích thước tối đa 2 GB) hoặc các thiết bị và hệ điều hành cũ không hỗ trợ các hệ thống tệp nâng cao.

Lệnh cơ bản nhất để tạo hệ thống tệp FAT là `mkfs.fat TARGET`, trong đó, `TARGET` là phân vùng cần hệ thống tệp được tạo trong đó (ví dụ như `mkfs.fat /dev/sdc1`).

Giống như các tiện ích khác, `mkfs.fat` cũng hỗ trợ một số tùy chọn dòng lệnh. Dưới đây là những tuỳ chọn quan trọng nhất. Danh sách đầy đủ và mô tả về mọi tùy chọn trong hướng dẫn sử dụng tiện ích sẽ có trong `man mkfs.fat`.

`-c`

Kiểm tra thiết bị đích để tìm các khối hỏng trước khi tạo hệ thống tệp.

`-C FILENAME BLOCK_COUNT`

Tạo tệp được chỉ định trong `FILENAME` và sau đó tạo một hệ thống tệp FAT bên trong tệp đó, tạo một "hình ảnh đĩa" trống. Tệp này sau này có thể được ghi vào thiết bị bằng tiện ích như `dd` hoặc được gắn kết dưới dạng vòng lặp thiết bị. Khi sử dụng tùy chọn này, số khối trong hệ thống tệp (`BLOCK_COUNT`) phải được chỉ định sau tên thiết bị.

`-F SIZE`

Chọn kích thước của FAT (*Bảng phân bổ Tệp*) trong khoảng 12, 16 hoặc 32, tức là giữa FAT12, FAT16 hoặc FAT32. Nếu không được chỉ định, `mkfs.fat` sẽ chọn tùy chọn thích hợp dựa trên kích thước hệ thống tệp.

`-n NAME`

Đặt nhãn ổ phân vùng hoặc tên cho hệ thống tệp. Tên này có thể dài tối đa 11 ký tự và mặc định là không có tên.

`-v`

Chế độ chi tiết. In nhiều thông tin hơn bình thường, hữu ích cho việc gỡ lỗi.

NOTE `mkfs.fat` *không thể* tạo hệ thống tệp "có thể khởi động". Theo trang hướng dẫn, "điều này không dễ như chúng ta nghĩ" và sẽ không được triển khai.

Tạo Hệ thống Tệp exFAT

exFAT là một hệ thống tệp do Microsoft tạo ra vào năm 2006 nhằm giải quyết một trong những hạn chế quan trọng nhất của FAT32 là kích thước tệp và đĩa. Trên exFAT, kích thước tệp tối đa là 16 exabyte (từ 4 GB trên FAT32) và kích thước đĩa tối đa là 128 petabyte.

Vì được hỗ trợ tốt bởi cả ba hệ điều hành chính (Windows, Linux và mac OS), đây là một lựa chọn lý tưởng khi người dùng cần khả năng tương tác, chẳng hạn như trên ổ đĩa flash dung lượng lớn, thẻ nhớ và ổ đĩa ngoại vi. Trên thực tế, đây là hệ thống tệp mặc định dành cho thẻ nhớ SDXC lớn hơn 32 GB như đã được *Hiệp Hội SD* xác định.

Tiện ích mặc định để tạo hệ thống tệp exFAT là `mkfs.exfat` - một liên kết đến `mkexfatfs`. Lệnh cơ bản nhất là `mkfs.exfat TARGET`, trong đó, `TARGET` là phân vùng nơi cần tạo hệ thống tệp (ví dụ như `mkfs.exfat /dev/sdb2`).

Trái ngược với các tiện ích khác được thảo luận trong bài học này, `mkfs.exfat` có rất ít tùy chọn dòng lệnh.

`-i VOL_ID`

Đặt ID ổ phân vùng thành giá trị được chỉ định trong `VOL_ID`. Đây là số thập lục phân 32 bit. Nếu không được xác định, ID dựa trên thời gian hiện tại sẽ được thiết lập

`-n NAME`

Đặt nhãn hoặc tên ổ phân vùng. Nó có thể có tối đa 15 ký tự và mặc định là không có tên.

`-p SECTOR`

Chỉ định cung đầu tiên của phân vùng đầu tiên trên đĩa. Đây là một giá trị tùy chọn và mặc định là 0.

`-s SECTORS`

Xác định số lượng cung vật lý trên mỗi cụm phân bổ. Nó phải là lũy thừa của hai, chẳng hạn như 1, 2, 4, 8, v.v.

Làm quen với Hệ thống Tệp Btrfs

Btrfs (được phát âm là “Butter FS”, “Better FS” hoặc thậm chí là “Butterfuss”, tên chính thức là *B-Tree Filesystem*) là một hệ thống tệp đã được phát triển từ năm 2007 dành riêng cho Linux bởi Tập đoàn Oracle và các công ty khác bao gồm Fujitsu, Red Hat, Intel, SUSE, v.v.

Có nhiều tính năng làm cho Btrfs trở nên hấp dẫn đối với các hệ thống hiện đại nơi dung lượng lưu trữ lớn là rất phổ biến. Những tính năng này gồm có hỗ trợ đa thiết bị (bao gồm phân đoạn, phản chiếu và phân phân đoạn+phản chiếu như trong thiết lập RAID), nén trong suốt, tối ưu hóa SSD, sao lưu gia tăng, chụp hình ảnh tức thời, chống phân mảnh trực tuyến, kiểm tra ngoại tuyến, hỗ trợ cho các ổ phân vùng con (có hạn ngạch), chống trùng lặp và nhiều hơn nữa.

Vì là hệ thống tệp *sao chép khi ghi* nên nó rất vững khi đứng trước các sự cố. Trên hết, Btrfs rất dễ sử dụng và được hỗ trợ tốt bởi nhiều bản phân phối Linux, một trong số chúng (như SUSE) còn sử dụng nó làm hệ thống tệp mặc định.

NOTE

Trên hệ thống tệp truyền thống, khi người dùng muốn ghi đè lên một phần của tệp, dữ liệu mới sẽ được đặt trực tiếp lên dữ liệu cũ mà nó đang thay thế. Trên hệ thống tệp *sao chép khi ghi*, dữ liệu mới sẽ được ghi vào không gian trống trên đĩa. Sau đó, siêu dữ liệu ban đầu của tệp sẽ được cập nhật để tham chiếu đến dữ liệu mới và chỉ khi đó dữ liệu cũ mới được giải phóng vì chúng không còn cần thiết nữa. Điều này sẽ làm giảm khả năng mất dữ liệu trong trường hợp xảy ra sự cố vì dữ liệu cũ chỉ bị loại bỏ sau khi hệ thống tệp hoàn toàn chắc chắn rằng nó không còn cần thiết nữa và dữ liệu mới đã được đưa vào.

Tạo Hệ thống Tệp Btrfs

Tiện ích `mkfs.btrfs` được sử dụng để tạo hệ thống tệp Btrfs. Nếu chúng ta sử dụng lệnh này mà không có bất kỳ tùy chọn nào thì nó sẽ tạo hệ thống tệp Btrfs trên một thiết bị nhất định giống như sau:

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

TIP Nếu không có tiện ích `mkfs.btrfs` trên hệ thống, hãy tìm `btrfs-progs` trong trình quản lý gói của bản phân phối.

Ta có thể sử dụng `-L` để đặt nhãn (hoặc tên) cho hệ thống tệp của mình. Nhãn btrfs có thể có tối đa 256 ký tự ngoại trừ ký tự xuống dòng:

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

TIP Hãy đặt nhãn trong dấu trích dẫn kép (như trên) nếu nhãn có khoảng trắng.

Có một lưu ý đặc biệt về Btrfs: ta có thể truyền *đa* thiết bị cho lệnh `mkfs.btrfs`. Truyền đa thiết bị sẽ mở rộng hệ thống tệp trên tất cả các thiết bị tương tự như thiết lập RAID hoặc LVM. Để chỉ định cách phân phối siêu dữ liệu trong mảng đĩa, hãy sử dụng tham số `-m`. Các tham số hợp lệ là `raid0`, `raid1`, `raid5`, `raid6`, `raid10`, `single` và `dup`.

Ví dụ: để tạo một hệ thống tệp mở rộng `/dev/sdb1` và `/dev/sdc1` và nối hai phân vùng thành một phân vùng lớn, hãy sử dụng:

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

WARNING Các hệ thống tệp mở rộng trên nhiều phân vùng như trên thoạt nhìn có vẻ có nhiều ưu thế nhưng nó lại không phải là một ý hay trên phương diện an toàn dữ liệu vì lỗi trên một đĩa đơn của mảng tương đương với việc mất một số dữ liệu nhất định. Rủi ro sẽ càng lớn khi ta sử dụng nhiều đĩa hơn vì khi đó sẽ có nhiều điểm có thể xảy ra lỗi hơn.

Quản lý Ổ phân vùng con

Ổ phân vùng con cũng giống như các hệ thống tệp ở bên trong hệ thống tệp. Hãy coi chúng như một thư mục có thể được gắn kết (và được xử lý) như một hệ thống tệp riêng biệt. Các ổ phân vùng con sẽ giúp việc tổ chức và quản trị hệ thống trở nên dễ dàng hơn vì mỗi ổ phân vùng con có thể có hạn ngạch hoặc quy tắc hình ảnh tức thời riêng.

NOTE Ổ phân vùng con không phải là phân vùng. Một phân vùng sẽ phân bổ một không gian cố định trên ổ đĩa. Điều này có thể dẫn đến các vấn đề liên đới như một phân vùng hết dung lượng trong khi phân vùng khác lại vẫn còn nhiều dung lượng. Với các ổ phân vùng con thì không như vậy vì chúng sẽ "chia sẻ`" dung lượng trống từ hệ thống tệp gốc của mình và mở rộng khi cần thiết.

Giả sử chúng ta có một hệ thống tệp Btrfs được gắn kết trên `/mnt/disk` và muốn tạo một ổ phân vùng con bên trong nó để lưu trữ các bản sao lưu của mình. Hãy gọi nó là `BKP`:

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

Tiếp theo, chúng ta sẽ liệt kê nội dung của hệ thống tệp `/mnt/disk`. Có thể thấy rằng đã có một thư mục mới được đặt tên theo ổ phân vùng con của chúng ta.

```
$ 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 Các ổ phân vùng con *cũng* có thể được truy cập giống như bất kỳ một thư mục nào khác.

Chúng ta có thể kiểm tra xem ổ phân vùng con có đang hoạt động hay không bằng lệnh:

```
# 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):
```

Ta có thể gắn ổ phân vùng con vào `/mnt/BKP` bằng cách truyền tham số `-t btrfs -o subvol=NAME` cho lệnh `mount`:

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

NOTE Tham số `-t` chỉ định loại hệ thống tệp sẽ được gắn kết.

Làm việc với Hình ảnh tức thời

Hình ảnh tức thời (Snapshots) cũng giống như ổ phân vùng con nhưng được điền sẵn nội dung từ đĩa nơi hình ảnh tức thời được chụp.

Khi được tạo, hình ảnh tức thời và ổ phân vùng gốc sẽ có nội dung hoàn toàn giống nhau. Nhưng từ thời điểm đó trở đi, nội dung của chúng sẽ hoàn toàn độc lập. Những thay đổi được thực hiện đối với ổ phân vùng gốc (như tệp được thêm, đổi tên hoặc xóa) sẽ không được phản ánh trên hình ảnh tức thời và ngược lại.

Hãy nhớ rằng hình ảnh tức thời sẽ *không* nhân bản các tệp và ban đầu hầu như sẽ không chiếm dung lượng đĩa. Nó chỉ đơn giản là sao chép cây hệ thống tệp trong khi trỏ đến dữ liệu gốc.

Lệnh tạo hình ảnh tức thời cũng giống như lệnh được sử dụng để tạo một ổ phân vùng con và chỉ cần thêm tham số `snapshot` sau `btrfs subvolume`. Lệnh bên dưới sẽ tạo một hình ảnh tức thời của hệ thống tệp Btrfs được gắn kết trong `/mnt/disk` trong `/mnt/disk/snap`:

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

Bây giờ, hãy tưởng tượng rằng chúng ta có những nội dung sau trong `/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
```

Hãy lưu ý thư mục snap chứa hình ảnh tức thời. Bây giờ, chúng ta sẽ xóa một số tệp và kiểm tra nội dung thư mục: `

```
$ 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
```

Tuy nhiên, nếu ta kiểm tra bên trong thư mục snap, các tệp đã xóa vẫn còn đó và có thể được khôi phục nếu cần.

```
$ 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
```

Chúng ta cũng có thể tạo hình ảnh tức thời chỉ cho phép đọc. Chúng cũng hoạt động y hệt như các hình ảnh tức thời có thể ghi, điểm khác biệt là nội dung của hình ảnh tức thời không thể thay đổi mà sẽ bị "đóng băng" theo thời gian. Ta chỉ cần thêm tham số `-r` khi tạo hình ảnh tức thời:

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

Vài dòng về tác vụ Nén

Btrfs có hỗ trợ tính năng nén tệp trong suốt với ba thuật toán khác nhau có sẵn. Tính năng này được thực hiện tự động theo từng tệp, miễn là hệ thống tệp được gắn kết với tùy chọn `-o compress`. Các thuật toán đủ thông minh để phát hiện ra các tệp không nén được và sẽ không cố nén chúng để giúp tiết kiệm tài nguyên hệ thống. Vì vậy, trên mỗi một thư mục, chúng ta có thể có cả các tệp nén và không nén. Thuật toán nén mặc định là ZLIB, nhưng ngoài ra còn có LZO (nhanh

hơn nhưng tỷ lệ nén kém hơn) hoặc ZSTD (nhanh hơn ZLIB với tỷ lệ nén tương đương) với nhiều cấp độ nén (xem các mục tiêu tương ứng trên các tùy chọn gắn kết).

Quản lý Phân vùng bằng GNU Parted

GNU Parted là một trình soạn thảo phân vùng rất mạnh (do đó mà nó có cái tên như vậy) có thể được sử dụng để tạo, xóa, di chuyển, thay đổi kích thước, giải cứu và sao chép phân vùng. Nó có thể hoạt động với cả đĩa GPT và MBR, đồng thời đáp ứng hầu hết mọi nhu cầu quản lý đĩa của người dùng.

Có nhiều giao diện người dùng đồ họa giúp ta làm việc với `parted` dễ dàng hơn rất nhiều như *GParted* dành cho môi trường máy tính để bàn dựa trên GNOME và *Trình quản lý Phân vùng KDE* dành cho Máy tính để bàn KDE. Tuy nhiên, người dùng nên học cách sử dụng `parted` trên dòng lệnh vì chúng ta không thể trông cậy vào việc có sẵn môi trường đồ hoạ máy tính để bàn trong cài đặt máy chủ.

WARNING

Không giống như `fdisk` và `gdisk`, `parted` sẽ thực hiện các thay đổi đối với đĩa *ngay lập tức* sau khi lệnh được đưa ra mà không cần đợi một lệnh khác ghi các thay đổi vào đĩa. Người dùng nên thực hành trên một đĩa hoặc ổ flash trống hoặc dự phòng để tránh nguy cơ mất dữ liệu nếu mắc lỗi.

Cách đơn giản nhất để bắt đầu sử dụng `parted` là nhập `parted DEVICE`, trong đó, `DEVICE` là thiết bị cần quản lý (`parted /dev/sdb`). Chương trình sẽ khởi động một giao diện dòng lệnh tương tác như `fdisk` và `gdisk` với dấu nhắc `(parted)` để nhập lệnh.

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

(parted)
```

WARNING

Hãy thật cẩn trọng! Nếu bạn không chỉ định một thiết bị, `parted` sẽ tự động chọn đĩa chính (thường là `/dev/sda`) để làm việc.

Chọn Đĩa

Để chuyển sang một đĩa khác với đĩa được chỉ định trên dòng lệnh, ta có thể sử dụng lệnh `select`, theo sau là tên của thiết bị:

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

```
Using /dev/sdb
```

Nhận Thông tin

Lệnh `print` có thể được sử dụng để lấy thêm thông tin về một phân vùng cụ thể hoặc thậm chí là về tất cả các thiết bị khối (đĩa) được kết nối với hệ thống.

Để lấy thông tin về phân vùng hiện được chọn, ta chỉ cần gõ `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)
```

Chúng ta có thể nhận danh sách tất cả các thiết bị khối được kết nối với hệ thống bằng cách sử dụng `print devices`:

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

Để nhận thông tin về tất cả các thiết bị được kết nối cùng một lúc, ta có thể sử dụng `print all`. Nếu muốn biết có bao nhiêu dung lượng trống trong mỗi thiết bị, ta có thể sử dụng `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
```

Tạo Bảng Phân vùng trên Đĩa trống

Để tạo bảng phân vùng trên một đĩa trống, hãy sử dụng lệnh `mklabel`, theo sau là loại bảng phân vùng cần sử dụng.

Có nhiều loại bảng phân vùng được hỗ trợ, nhưng các loại chính chúng ta nên biết là `msdos` được sử dụng để chỉ bảng phân vùng MBR và `gpt` để chỉ bảng phân vùng GPT. Để tạo bảng phân vùng MBR, hãy nhập:

```
(parted) mklabel msdos
```

Và để tạo bảng phân vùng GPT, lệnh sẽ là:

```
(parted) mklabel gpt
```

Tạo một Phân vùng

Để tạo một phân vùng, ta sử dụng lệnh `mkpart` với cú pháp `mkpart PARTTYPE FSTYPE START END`, trong đó:

`PARTTYPE`

Loại phân vùng, có thể là `primary` (chính), `logical` (logic) hoặc `extended` (mở rộng) trong trường hợp bảng phân vùng MBR được sử dụng.

`FSTYPE`

Chỉ định hệ thống tệp nào sẽ được sử dụng trên phân vùng này. Hãy lưu ý rằng `parted` sẽ *không* tạo hệ thống tệp. Nó chỉ đặt một cờ trên phân vùng để hệ điều hành biết sẽ có loại dữ liệu nào từ nó.

`START`

Chỉ định điểm chính xác trên thiết bị mà phân vùng bắt đầu. Ta có thể sử dụng các đơn vị khác nhau để chỉ định điểm này. `2s` có thể được dùng để chỉ cung thứ hai của đĩa, trong khi `1m` là để chỉ phần đầu của megabyte đầu tiên của đĩa. Các đơn vị phổ biến khác là `B` (byte) và `%` (phần trăm của đĩa).

`END`

Chỉ định phần cuối của phân vùng. Hãy lưu ý rằng đây *không* phải là kích thước của phân vùng mà là *điểm trên đĩa mà nó kết thúc*. Ví dụ: nếu ta chỉ định `100m` thì phân vùng sẽ đi đến điểm 100 MB sau khi khởi động đĩa. Ta có thể sử dụng các đơn vị giống như trong tham số `START`.

Vì vậy, lệnh

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

sẽ tạo một phân vùng chính thuộc loại `ext4` bắt đầu từ megabyte đầu tiên của đĩa và kết thúc sau megabyte thứ 100.

Xóa Phân vùng

Để xóa một phân vùng, hãy sử dụng lệnh `rm`, theo sau là số phân vùng mà ta có thể hiển thị bằng lệnh `print`. Vì vậy, `rm 2` sẽ xóa phân vùng thứ hai trên đĩa hiện được chọn.

Khôi phục Phân vùng

`parted` có thể khôi phục phân vùng đã bị xóa. Giả sử chúng ta có cấu trúc phân vùng sau:

```
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
```

Phân vùng 2 đã tình cờ bị xoá bằng `rm 2`. Để khôi phục nó, chúng ta có thể sử dụng lệnh `rescue` với cú pháp `rescue START END`, trong đó, `START` là vị trí gần đúng nơi phân vùng bắt đầu và `END` là vị trí gần đúng nơi phân vùng kết thúc.

`parted` sẽ quét đĩa để tìm kiếm các phân vùng và đề nghị khôi phục mọi phân vùng được tìm thấy. Trong ví dụ trên, phân vùng 2 bắt đầu ở 99,6 MB và kết thúc ở 200 MB. Vì vậy, chúng ta có thể sử dụng lệnh sau để khôi phục phân vùng:

```
(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
```

Lệnh này sẽ phục hồi phân vùng và nội dung của nó. Hãy lưu ý rằng `rescue` chỉ có thể khôi phục các phân vùng đã được cài đặt hệ thống tệp trên đó. Phân vùng trống sẽ không được phát hiện.

Thay đổi Kích thước Phân vùng ext2/3/4

`parted` có thể được sử dụng để thay đổi kích thước các phân vùng. Tuy nhiên, có một số lưu ý:

- Trong quá trình thay đổi kích thước, phân vùng phải được ngắt kết nối và không được sử dụng.
- Cần có đủ dung lượng trống *ở sau* phân vùng để điều chỉnh nó theo kích thước mong muốn.

Lệnh sẽ là `resizepart`, theo sau là số phân vùng và nơi kết thúc. Ví dụ: nếu chúng ta có bảng phân vùng sau:

```
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
```

Việc điều chỉnh kích thước phân vùng `1` bằng cách sử dụng `resizepart` sẽ gây ra thông báo lỗi vì với kích thước mới, phân vùng `1` sẽ chồng lên phân vùng `2`. Tuy nhiên, phân vùng `3` có thể được thay đổi kích thước vì có dung lượng trống phía sau nó. Việc này có thể được xác minh bằng lệnh `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
```

Vì vậy, ta có thể sử dụng lệnh sau để thay đổi kích thước phân vùng 3 thành 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
```

Hãy nhớ rằng điểm cuối mới được chỉ định tính từ điểm bắt đầu của đĩa. Do đó, vì phân vùng `3` đã kết thúc ở mức 300 MB nên bây giờ nó cần kết thúc ở mức 350 MB.

Tuy nhiên, việc thay đổi kích thước phân vùng chỉ là một phần của tác vụ. Chúng ta cũng cần thay đổi kích thước của hệ thống tệp nằm trong đó nữa. Đối với hệ thống tệp ext2/3/4, việc này có thể được thực hiện bằng lệnh `resize2fs`. Trong trường hợp ví dụ trên, phân vùng 3 vẫn hiển thị kích thước "cũ" khi được gắn kết:

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

Để điều chỉnh kích thước, chúng ta có thể sử dụng lệnh `resize2fs DEVICE SIZE`, trong đó, `DEVICE` tương ứng với phân vùng ta muốn thay đổi kích thước và `SIZE` là kích thước mới. Nếu bỏ qua tham số kích thước, lệnh sẽ sử dụng toàn bộ dung lượng có sẵn của phân vùng. Trước khi thay đổi kích thước, chúng ta nên ngắt gắn kết phân vùng.

Trong ví dụ trên:

```
$ 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
```

Để *thu hẹp* một phân vùng, tiến trình cần được thực hiện theo thứ tự ngược lại. *Đầu tiên*, ta sẽ thay

đổi kích thước hệ thống tệp thành kích thước mới nhỏ hơn, sau đó thay đổi kích thước của chính phân vùng đó bằng cách sử dụng `parted`.

WARNING Hãy chú ý khi thu hẹp phân vùng. Nếu làm sai thứ tự, bạn sẽ bị mất dữ liệu!

Trong ví dụ của chúng ta:

```
# 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 Thay vì chỉ định kích thước mới, ta có thể sử dụng tham số `-M` của `resize2fs` để điều chỉnh kích thước của hệ thống tệp sao cho nó vừa đủ lớn cho các tệp.

Tạo Phân vùng Hoán đổi

Trên Linux, hệ thống có thể hoán đổi các trang bộ nhớ từ RAM sang đĩa khi cần. Việc lưu trữ chúng trên một dung lượng riêng biệt thường được triển khai dưới dạng một phân vùng riêng trên một đĩa được gọi là phân vùng *hoán đổi* hoặc đơn giản là *hoán đổi*. Phân vùng này cần phải thuộc một loại cụ thể và được thiết lập với tiện ích thích hợp (`mkswap`) trước khi có thể được sử dụng.

Để tạo một phân vùng hoán đổi bằng cách sử dụng `fdisk` hoặc `gdisk`, chúng ta chỉ cần tiến hành

giống như khi tạo một phân vùng thông thường như đã được giải thích ở trên. Sự khác biệt duy nhất là ta cần thay đổi loại phân vùng thành *Linux swap*.

- Trên `fdisk`, hãy sử dụng lệnh `t`, chọn phân vùng muốn sử dụng và thay đổi loại của nó thành `82`, ghi các thay đổi vào đĩa và thoát bằng `w`.
- Trên `gdisk`, lệnh thay đổi loại phân vùng cũng là `t` nhưng mã lại là `8200`. Sau khi ghi các thay đổi vào đĩa, hãy thoát bằng `w`.

Nếu đang sử dụng `parted`, phân vùng đó phải được xác định là phân vùng hoán đổi trong quá trình tạo. Ta chỉ cần sử dụng `linux-swap` làm loại hệ thống tệp. Ví dụ: lệnh tạo phân vùng hoán đổi 500 MB, bắt đầu từ 300 MB trên đĩa sẽ là:

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

Sau khi phân vùng được tạo và xác định chính xác, chúng ta chỉ cần sử dụng `mkswap`, theo sau là thiết bị đại diện cho phân vùng muốn sử dụng, chẳng hạn như:

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

Để làm tính năng hoán đổi trên phân vùng này khả dụng, hãy sử dụng `swapon`, theo sau là tên thiết bị:

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

Tương tự, `swapoff` đi cùng với tên thiết bị sẽ tắt tính năng hoán đổi trên thiết bị đó.

Linux cũng hỗ trợ sử dụng *tệp* hoán đổi thay vì phân vùng hoán đổi. Ta chỉ cần tạo một tệp trống có kích thước tuỳ thích bằng cách sử dụng `dd`, sau đó sử dụng `mkswap` và `swapon` với tệp này làm mục tiêu.

Các lệnh sau sẽ tạo một tệp 1 GB có tên `myswap` trong thư mục hiện tại chứa đầy các số 0, sau đó thiết lập và kích hoạt tệp dưới dạng tệp hoán đổi.

Tạo tệp hoán đổi:

```
$ 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=` là tệp đầu vào, nguồn dữ liệu sẽ được ghi vào tệp này. Trong trường hợp này sẽ là thiết bị `/dev/zero` có thể cung cấp bao nhiêu ký tự `NULL` theo yêu cầu cũng được. `of=` là tệp đầu ra, tức tệp sẽ được tạo. `bs=` là kích thước của các khối dữ liệu (ở đây được chỉ định bằng Megabyte) và `count=` là số lượng khối được ghi vào đầu ra. 1024 khối 1 MB sẽ bằng một 1 GB.

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

# swapon myswap
```

Bằng cách sử dụng các lệnh trên, tệp hoán đổi này sẽ chỉ được sử dụng trong phiên hệ thống hiện tại. Nếu khởi động lại máy, tệp vẫn sẽ có sẵn nhưng sẽ không được tải tự động. Ta có thể tự động hóa việc này bằng cách thêm tệp hoán đổi mới vào `/etc/fstab`. Vấn đề này sẽ được thảo luận trong bài học sau.

TIP

Cả `mkswap` và `swapon` đều sẽ khiếu nại nếu tệp hoán đổi có các quyền không an toàn. Cờ quyền cho tệp được đề xuất là `0600`. Chủ sở hữu và nhóm phải là `root`.

Bài tập Hướng dẫn

1. Nên sử dụng lược đồ phân vùng nào để phân vùng một ổ cứng 3TB thành ba phân vùng 1 GB? Tại sao?

2. Trên `gdisk`, làm cách nào để có thể biết đĩa còn bao nhiêu dung lượng trống?

3. Lệnh tạo hệ thống tệp ext3, kiểm tra các khối hỏng trước đó (với nhãn `MyDisk` và UUID ngẫu nhiên) trên thiết bị `/dev/sdc1` là gì?

4. Bằng cách sử dụng `parted`, lệnh tạo phân vùng ext4 300 MB bắt đầu từ 500 MB trên đĩa là gì?

5. Hãy tưởng tượng bạn có 2 phân vùng, một trên `/dev/sda1` và một trên `/dev/sda2`, cả hai đều có kích thước 20 GB. Làm cách nào để có thể sử dụng chúng trên một hệ thống tệp Btrfs duy nhất theo cách mà nội dung của một phân vùng sẽ tự động được phản chiếu trên phân vùng còn lại giống như trên thiết lập RAID1? Hệ thống tệp sẽ lớn đến mức nào?

Bài tập Mở rộng

1. Hãy xem xét một đĩa 2 GB có bảng phân vùng MBR và cách bố trí sau:

```
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
```

   Bạn có thể tạo phân vùng 600 MB trên đó không? Tại sao?

2. Trên đĩa tại `/dev/sdc`, chúng ta có phân vùng đầu tiên là 1 GB chứa khoảng 256 MB tệp. Bằng cách sử dụng `parted`, bạn sẽ thu hẹp nó như thế nào để nó có đủ dung lượng cho các tệp?

3. Hãy tưởng tượng bạn có một ổ đĩa tại `/dev/sdb` và bạn muốn tạo một phân vùng hoán đổi 1 GB tại điểm bắt đầu của phân vùng đó. Vì vậy, bằng cách sử dụng `parted`, bạn đã tạo phân vùng với `mkpart primary linux-swap 0 1024M`. Sau đó, bạn kích hoạt tính năng hoán đổi trên phân vùng này với `swapon /dev/sdb1` nhưng nhận được thông báo lỗi sau:

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

   Lỗi là do đâu?

4. Xuyên suốt bài học này, bạn đã thử một số lệnh trong `parted`. Nhưng do nhầm lẫn nên bạn đã xóa phân vùng thứ 3 trên ổ cứng. Bạn biết rằng nó xuất hiện sau phân vùng UEFI 250 MB và phân vùng hoán đổi 4 GB và có kích thước 10 GB. Bạn có thể sử dụng lệnh nào để khôi phục nó?

5. Hãy tưởng tượng bạn có một phân vùng 4 GB chưa sử dụng trên `/dev/sda3`. Bằng cách sử dụng `fdisk`, trình tự thao tác sẽ như thế nào để biến nó thành một phân vùng được kích hoạt tính năng hoán đổi?

Tóm tắt

Trong bài học này, chúng ta đã học về:

- Cách tạo bảng phân vùng MBR trên đĩa bằng `fdisk` và cách sử dụng nó để tạo và xóa phân vùng.
- Cách tạo bảng phân vùng MBR trên đĩa bằng `gdisk` và cách sử dụng nó để tạo và xóa phân vùng.
- Cách tạo phân vùng ext2, ext3, ext4, XFS, VFAT và exFAT.
- Cách sử dụng `parted` để tạo, xóa và khôi phục phân vùng trên cả đĩa MBR và GPT.
- Cách thay đổi kích thước phân vùng ext2, ext3, ext4 và Brts.
- Cách tạo, thiết lập và kích hoạt phân vùng hoán đổi và tệp hoán đổi.

Các lệnh sau đã được thảo luận trong bài học này:

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

Đáp án Bài tập Hướng dẫn

1. Nên sử dụng lược đồ phân vùng nào để phân vùng một ổ cứng 3TB thành ba phân vùng 1 GB? Tại sao?

   GPT, vì MBR hỗ trợ tối đa 2TB ổ cứng.

2. Trên `gdisk`, làm cách nào để có thể biết đĩa còn bao nhiêu dung lượng trống?

   Sử dụng `p` (in). Tổng dung lượng trống sẽ được hiển thị dưới dạng dòng thông tin cuối cùng trước bảng phân vùng.

3. Lệnh tạo hệ thống tệp ext3, kiểm tra các khối hỏng trước đó (với nhãn `MyDisk` và UUID ngẫu nhiên) trên thiết bị `/dev/sdc1` là gì?

   Lệnh sẽ là `mkfs.ext3 -c -L MyDisk -U random /dev/sdc1`. Ngoài ra, `mke2fs -t ext3` cũng có thể được sử dụng thay vì `mkfs.ext3`

4. Bằng cách sử dụng `parted`, lệnh tạo phân vùng ext4 300 MB bắt đầu từ 500 MB trên đĩa là gì?

   Sử dụng `mkpart chính ext4 500m 800m`. Hãy nhớ rằng bạn sẽ phải tạo hệ thống tệp bằng cách sử dụng `mkfs.ext4` vì parted không làm được điều này.

5. Hãy tưởng tượng bạn có 2 phân vùng, một trên `/dev/sda1` và một trên `/dev/sda2`, cả hai đều có kích thước 20 GB. Làm cách nào để có thể sử dụng chúng trên một hệ thống tệp Btrfs duy nhất theo cách mà nội dung của một phân vùng sẽ tự động được phản chiếu trên phân vùng còn lại giống như trên thiết lập RAID1? Hệ thống tệp sẽ lớn đến mức nào?

   Sử dụng `mkfs.btrfs /dev/sda1 /dev/sdb1 -m raid1`. Hệ thống tệp kết quả sẽ có kích thước 20 GB vì một phân vùng sẽ hoạt động đơn thuần như một bản sao của phân vùng còn lại.

Đáp án Bài tập Mở rộng

1. Hãy xem xét một đĩa 2 GB có bảng phân vùng MBR và cách bố trí sau:

```
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
```

   Bạn có thể tạo phân vùng 600 MB trên đó không? Tại sao?

   Không thể, lý do là vì không có đủ dung lượng liền kề. Manh mối đầu tiên cho thấy có vấn đề gì đó "không ổn" là danh sách thiết bị: bạn có `/dev/sdb1` và `/dev/sdb3` nhưng không có `/dev/sdb2`. Điều này có nghĩa là đang thiếu thiết bị.

   Sau đó, bạn cần xem nơi một phân vùng kết thúc và phân vùng khác bắt đầu. Phân vùng 1 kết thúc tại cung số `1050623` và phân vùng 2 bắt đầu tại `2099200`. Đó là một "khoảng trống" gồm 1048577 cung. Với mỗi cung là 512 byte, khoảng trống đó sẽ là 536.871.424 byte. Nếu bạn chia nó cho 1024, bạn sẽ nhận được 524,288 Kilobyte. Chia lại cho 1024 và bạn nhận được... 512 MB. Đây là kích thước của "khoảng trống".

   Nếu đĩa có dung lượng là 2 GB thì chúng ta có tối đa 512 MB sau phân vùng 3. Ngay cả khi ta có tổng cộng khoảng 1 GB chưa được phân bổ thì khối liền kề lớn nhất vẫn là 512 MB. Vì vậy, ta sẽ không có dung lượng cho một phân vùng 600 MB.

2. Trên đĩa tại `/dev/sdc`, chúng ta có phân vùng đầu tiên là 1 GB chứa khoảng 256 MB tệp. Bằng cách sử dụng `parted`, bạn sẽ thu hẹp nó như thế nào để nó có đủ dung lượng cho các tệp?

   Đây là một hoạt động gồm nhiều phần. Trước tiên, bạn phải thu nhỏ hệ thống tệp bằng cách sử dụng `resize2fs`. Thay vì chỉ định trực tiếp kích thước mới, bạn có thể sử dụng tham số `-M` để nó chỉ "vừa đủ lớn". Vì vậy nên ta có `resize2fs -M /dev/sdc1`.

   Sau đó, bạn sẽ thay đổi kích thước phân vùng bằng cách sử dụng `resizepart`. Vì đây là phân vùng đầu tiên nên chúng ta có thể giả sử rằng nó bắt đầu ở 0 và kết thúc ở 241 MB. Vì vậy, lệnh sẽ là `resizepart 1 241M`.

3. Hãy tưởng tượng bạn có một ổ đĩa tại `/dev/sdb` và bạn muốn tạo một phân vùng hoán đổi 1 GB tại điểm bắt đầu của phân vùng đó. Vì vậy, bằng cách sử dụng `parted`, bạn đã tạo phân vùng với `mkpart primary linux-swap 0 1024M`. Sau đó, bạn kích hoạt tính năng hoán đổi trên phân vùng này với `swapon /dev/sdb1` nhưng nhận được thông báo lỗi sau:

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

   Lỗi là do đâu?

   Bạn đã tạo một phân vùng đúng loại (`linux-swap`), nhưng hãy nhớ rằng `mkpart` sẽ *không tạo hệ thống tệp*. Bạn đã quên thiết lập phân vùng dưới dạng dung lượng hoán đổi với `mkswap` trước khi sử dụng.

4. Xuyên suốt bài học này, bạn đã thử một số lệnh trong `parted`. Nhưng do nhầm lẫn nên bạn đã xóa phân vùng thứ 3 trên ổ cứng. Bạn biết rằng nó xuất hiện sau phân vùng UEFI 250 MB và phân vùng hoán đổi 4 GB và có kích thước 10 GB. Bạn có thể sử dụng lệnh nào để khôi phục nó?

   Đừng hoảng hốt, bạn có đủ tất cả những thông tin cần thiết để khôi phục phân vùng. Bạn chỉ cần sử dụng `rescue` và thực hiện phép toán. Bạn đã có 250 MB + 4,096 MB (4*1024) trước đó. Do đó, điểm bắt đầu phải vào khoảng 4346 MB. Cộng với kích thước 10,240 MB (10*1024), nó sẽ kết thúc ở 14,586 MB. Vì vậy, `rescue 4346m 14586m` sẽ là câu trả lời. Bạn sẽ phải "thông cảm" cho lệnh rescue vì nó có thể bị chênh một chút ở điểm bắt đầu hoặc kết thúc, tùy thuộc vào hình dạng đĩa của bạn.

5. Hãy tưởng tượng bạn có một phân vùng 4 GB chưa sử dụng trên `/dev/sda3`. Bằng cách sử dụng `fdisk`, trình tự thao tác sẽ như thế nào để biến nó thành một phân vùng được kích hoạt tính năng hoán đổi?

   Đầu tiên, hãy thay đổi loại phân vùng thành "Linux Swap" (82), ghi các thay đổi của bạn vào đĩa và thoát. Sau đó, hãy sử dụng `mkswap` để thiết lập phân vùng làm vùng hoán đổi. Sau đó, hãy sử dụng `swapon` để kích hoạt nó.

104.2 Duy trì tính toàn vẹn của các Hệ thống Tệp

Tham khảo các mục tiêu LPI

LPIC-1 version 5.0, Exam 101, Objective 104.2

Khối lượng

2

Các lĩnh vực kiến thức chính

- Xác minh tính toàn vẹn của hệ thống tệp.
- Giám sát không gian trống và inode.
- Sửa chữa các sự cố hệ thống tệp đơn giản.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

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

Linux Professional Institute

104.2 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	104 Thiết bị, Hệ thống tệp Linux, Tiêu chuẩn phân cấp hệ thống Tệp
Mục tiêu:	104.2 Duy trì tính toàn vẹn của Hệ thống Tệp
Bài:	1 trên 1

Giới thiệu

Các hệ thống tệp Linux hiện đại đều có chức năng ghi nhật ký. Điều này có nghĩa là mọi thao tác vận hành đều sẽ được ghi vào nhật ký nội bộ (*journal*) trước khi được thực thi. Nếu hoạt động bị gián đoạn do lỗi hệ thống (như lỗi hạt nhân, mất điện, v.v.), nó có thể được tái thiết bằng cách kiểm tra nhật ký để tránh việc hư hỏng hệ thống tệp và mất dữ liệu.

Điều này làm giảm đáng kể nhu cầu kiểm tra hệ thống tệp thủ công (dù chúng ta có thể vẫn sẽ cần đến chúng). Sự khác biệt giữa việc hiểu và không hiểu rõ về các công cụ được sử dụng cho tác vụ này (và các thông số tương ứng) cũng giống như việc dùng bữa tối ở nhà với gia đình so với việc thức trắng đêm làm việc trong phòng máy.

Trong bài học này, chúng ta sẽ thảo luận về các công cụ có sẵn để giám sát việc sử dụng hệ thống tệp, tối ưu hóa hoạt động của nó cũng như cách kiểm tra và sửa chữa hư hỏng.

Kiểm tra việc Sử dụng Đĩa

Có hai lệnh có thể được sử dụng để kiểm tra lượng dung lượng đang được sử dụng và còn lại trên

hệ thống tệp. Lệnh đầu tiên là `du`, viết tắt của “disk usage”.

`du` có tính chất đệ quy. Ở dạng cơ bản nhất, lệnh này sẽ chỉ hiển thị số lượng khối 1 Kilobyte đang được sử dụng bởi thư mục hiện tại và tất cả các thư mục con của nó:

```
$ du
4816    .
```

Kết quả có vẻ không được hữu ích cho lắm. Vì vậy, chúng ta có thể yêu cầu thêm đầu ra ở dạng “con người có thể đọc được” bằng cách thêm tham số `-h`:

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

Theo mặc định, `du` sẽ chỉ hiển thị số lượng sử dụng cho các thư mục (bao gồm tất cả các tệp và thư mục con bên trong nó). Để hiển thị số lượng riêng lẻ cho tất cả các tệp trong thư mục, hãy sử dụng tham số `-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    .
```

Hành vi mặc định sẽ hiển thị mức sử dụng của mọi thư mục con, sau đó là tổng mức sử dụng của thư mục hiện tại *bao gồm* cả các thư mục con:

```
$ du -h
```

```
4.8M    ./Temp
6.0M    .
```

Trong ví dụ trên, chúng ta có thể thấy rằng thư mục con `Temp` chiếm 4,8 MB và thư mục hiện tại (*bao gồm* `Temp`) chiếm 6,0 MB. Nhưng *các tệp* trong thư mục hiện tại chiếm bao nhiêu dung lượng nếu không bao gồm các thư mục con? Để biết được điều này, chúng ta phải sử dụng tham số `-S`:

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

TIP Hãy nhớ rằng các tham số dòng lệnh có phân biệt chữ hoa chữ thường: `-s` sẽ khác với `-S`.

Nếu vẫn muốn tách biệt giữa dung lượng được sử dụng bởi các tệp trong thư mục hiện tại và dung lượng được sử dụng bởi các thư mục con, thêm vào đó là *cả* số tổng dung lượng ở cuối, ta có thể thêm tham số `-c`:

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

Chúng ta có thể kiểm soát mức độ "sâu" của đầu ra của `du` với tham số `-d N`, trong đó, `N` sẽ mô tả các cấp độ. Ví dụ: nếu ta sử dụng tham số `-d 1`, nó sẽ hiển thị thư mục hiện tại và các thư mục con của nó nhưng sẽ không hiển thị các thư mục con của những thư mục đó.

Hãy xem sự khác biệt bên dưới. Khi không có `-d`:

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

Và giới hạn độ sâu ở một cấp độ với `-d 1`:

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

Hãy lưu ý rằng ngay cả khi `anotherdir` không được hiển thị, kích thước của nó vẫn sẽ được tính đến.

Chúng ta có thể sẽ muốn loại trừ một số loại tệp với `--exclude="PATTERN"`, trong đó, `PATTERN` là mẫu cần loại trừ. Hãy xem thử thư mục này:

```
$ 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    .
```

Bây giờ, chúng ta sẽ sử dụng `--exclude` để lọc tất cả các tệp có phần mở rộng `.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    .
```

Hãy lưu ý rằng tổng số dung lượng bây giờ không còn phản ánh kích thước của các tệp bị loại trừ.

Kiểm tra Dung lượng trống

`du` làm việc ở cấp độ tệp. Có một lệnh khác có thể hiển thị mức sử dụng đĩa và dung lượng sẵn có ở cấp hệ thống tệp là `df`.

Lệnh `df` sẽ cung cấp danh sách tất cả các hệ thống tệp có sẵn (đã được gắn kết) trên hệ thống bao gồm tổng kích thước của chúng, dung lượng đã được sử dụng, dung lượng còn trống, tỷ lệ phần trăm sử dụng và nơi nó được gắn kết:

```
$ 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
```

Tuy nhiên, việc hiển thị kích thước theo khối 1 KB không được thân thiện với người dùng cho lắm. Giống như trên `du`, chúng ta có thể thêm các tham số `-h` để có được kết quả đầu ra "dễ đọc hơn":

```
$ 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
```

Ta cũng có thể sử dụng tham số `-i` để hiển thị các nút định danh đã sử dụng/có sẵn thay vì các khối:

```
$ 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
```

Có một tham số hữu ích là `-T`. Tham số này cũng sẽ in loại của từng hệ thống tệp:

```
$ 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
```

Khi đã biết loại hệ thống tệp, chúng ta có thể lọc đầu ra. Ta chỉ có thể hiển thị các hệ thống tệp thuộc cùng một loại đã cho bằng `-t TYPE` hoặc loại trừ các hệ thống tệp thuộc một loại nhất định bằng `-x TYPE` như trong các ví dụ bên dưới đây.

Không bao gồm hệ thống tệp `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
```

Chỉ hiển thị hệ thống tệp `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
```

Ta cũng có thể tùy chỉnh đầu ra của `df` và chọn nội dung sẽ được hiển thị cũng như theo thứ tự nào bằng cách sử dụng tham số `--output=`, theo sau là danh sách các trường được phân tách bằng dấu phẩy cần hiển thị. Một số trường có sẵn là:

`source`
: Thiết bị tương ứng với hệ thống tệp.

`fstype`
: Loại hệ thống tệp.

`size`
: Tổng kích thước của hệ thống tệp.

`used`
: Lượng dung lượng đang được sử dụng.

`avail`
: Lượng dung lượng có sẵn.

`pcent`
: Tỷ lệ sử dụng.

`target`
: Nơi hệ thống tệp được gắn kết (điểm gắn kết).

Nếu muốn một đầu ra hiển thị điểm mục tiêu, nguồn, loại và tỷ lệ sử dụng, chúng ta có thể sử dụng:

```
$ 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%
```

`df` cũng có thể được sử dụng để kiểm tra thông tin nút định danh bằng cách chuyển các trường sau tới `--output=`:

`itotal`

Tổng số nút định danh trong hệ thống tệp.

`iused`

Số lượng nút định danh được sử dụng trong hệ thống tệp.

`iavail`

Số lượng nút định danh có sẵn trong hệ thống tệp.

`ipcent`

Tỷ lệ phần trăm các nút định danh được sử dụng trong hệ thống tệp.

Ví dụ:

```
$ 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%
```

Duy trì Hệ thống Tệp ext2, ext3 và ext4

Để kiểm tra lỗi hệ thống tệp (và sửa được chúng), Linux có cung cấp tiện ích `fsck` (hãy nghĩ đến "filesystem check" và bạn sẽ không bao giờ quên tên lệnh). Ở dạng cơ bản nhất, chúng ta có thể gọi nó bằng `fsck`, theo sau là vị trí của hệ thống tệp cần kiểm tra:

```
# 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 KHÔNG BAO GIỜ chạy `fsck` (hoặc các tiện ích liên quan) trên một hệ thống tệp đã được gắn kết. Nếu vẫn cố thực hiện thao tác này, dữ liệu có thể sẽ bị mất.

Bản thân `fsck` sẽ không kiểm tra hệ thống tệp mà sẽ chỉ gọi tiện ích thích hợp cho loại hệ thống tệp để thực hiện việc đó. Trong ví dụ trên, do loại hệ thống tệp không được chỉ định nên `fsck` đã giả định hệ thống tệp ext2/3/4 theo mặc định và gọi `e2fsck`.

Để chỉ định một hệ thống tệp, hãy sử dụng tùy chọn `-t`, theo sau là tên hệ thống tệp (như trong `fsck -t vfat /dev/sdc`). Ngoài ra, ta có thể gọi trực tiếp một tiện ích dành riêng cho hệ thống tệp (như `fsck.msdos` cho hệ thống tệp FAT).

TIP Nhập `fsck` theo sau là `Tab` hai lần để xem danh sách tất cả các lệnh liên quan trên hệ thống.

`fsck` có thể nhận một số đối số dòng lệnh. Sau đây là một trong số những đối số phổ biến nhất:

`-A`
: Kiểm tra tất cả các hệ thống tệp được liệt kê trong `/etc/fstab`.

`-C`
: Hiển thị thanh tiến trình khi kiểm tra hệ thống tệp. Hiện tại nó chỉ hoạt động trên hệ thống tệp ext2/3/4.

`-N`
: In những gì sẽ được thực hiện và thoát ra mà không thực sự kiểm tra hệ thống tệp.

`-R`
: Khi được sử dụng cùng với `-A`, tham số này sẽ bỏ qua việc kiểm tra hệ thống tệp gốc.

`-V`
: Chế độ chi tiết, in nhiều thông tin hơn bình thường trong quá trình hoạt động. Tham số này rất hữu ích cho việc gỡ lỗi.

Tiện ích cụ thể cho các hệ thống tệp ext2, ext3 và ext4 là `e2fsck` hay còn được gọi là `fsck.ext2`, `fsck.ext3` và `fsck.ext4` (ba tiện ích này chỉ đơn thuần là liên kết đến `e2fsck`). Theo mặc định, nó sẽ chạy ở chế độ tương tác: khi tìm thấy lỗi hệ thống tệp, nó sẽ dừng lại và hỏi người dùng xem phải làm gì. Người dùng phải nhập `y` để khắc phục sự cố, `n` để giữ nguyên không sửa hoặc `a` để

khắc phục sự cố hiện tại và tất cả các sự cố tiếp theo.

Tất nhiên việc ngồi chờ cho `e2fsck` hỏi ta phải làm gì không phải là một cách sử dụng thời gian hiệu quả, đặc biệt là khi chúng ta đang xử lý một hệ thống tệp lớn. Vì vậy, `e2fsck` có các tùy chọn để chạy ở chế độ không tương tác:

`-p`
: Tuỳ chọn này sẽ tự động sửa bất kỳ lỗi nào được tìm thấy. Nếu tìm thấy một lỗi cần tới quản trị viên hệ thống can thiệp, `e2fsck` sẽ cung cấp mô tả về sự cố và thoát.

`-y`
: Tuỳ chọn này sẽ trả lời `y` (yes) cho tất cả các câu hỏi.

`-n`
: Ngược lại với `-y`. Ngoài việc trả lời `n` (no) cho tất cả các câu hỏi, tuỳ chọn này sẽ khiến hệ thống tệp được gắn kết ở chế độ chỉ đọc nên sẽ không thể bị sửa đổi.

`-f`
: Buộc `e2fsck` kiểm tra hệ thống tệp ngay cả khi nó được đánh dấu là “clean” (sạch), tức là đã được ngắt gắn kết chính xác.

Tinh chỉnh Hệ thống Tệp ext

Hệ thống tệp ext2, ext3 và ext4 có một số tham số có thể được quản trị viên hệ thống điều chỉnh hoặc "tinh chỉnh" (tune) để phù hợp hơn với nhu cầu của hệ thống. Tiện ích được sử dụng để hiển thị hoặc sửa đổi các tham số này được gọi là `tune2fs`.

Để xem xét các tham số hiện tại cho bất kỳ hệ thống tệp cụ thể nào, hãy sử dụng tham số `-l`, theo sau là thiết bị đại diện cho phân vùng. Ví dụ bên dưới sẽ hiển thị đầu ra của lệnh này trên phân vùng đầu tiên của đĩa đầu tiên (`/dev/sda1`) của máy:

```
# 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
```

Hệ thống tệp ext có chức năng *đếm số lượng gắn kết*. Số lượng sẽ được tăng thêm 1 mỗi lần hệ thống tệp được gắn kết và khi đạt đến giá trị ngưỡng (*số lượng gắn kết tối đa*), hệ thống sẽ tự động

được kiểm tra bằng `e2fsck` trong lần khởi động tiếp theo.

Số lượng gắn kết tối đa có thể được đặt bằng tham số `-c N`, trong đó, `N` là số lần hệ thống tệp có thể được gắn kết mà không bị kiểm tra. Tham số `-C N` sẽ đặt số lần hệ thống được gắn kết vào giá trị `N`. Hãy lưu ý rằng các tham số dòng lệnh có phân biệt chữ hoa và chữ thường, vì thế nên `-c` sẽ khác với `-C`.

Chúng ta cũng có thể xác định khoảng thời gian giữa các lần kiểm tra với tham số `-i`, theo sau là một số và các chữ cái `d` cho ngày, `m` cho tháng và `y` cho năm. Ví dụ: `-i 10d` sẽ kiểm tra hệ thống tệp ở lần khởi động lại tiếp theo sau mỗi 10 ngày. Ta có thể sử dụng giá trị 0 để tắt tính năng này.

`-L` có thể được sử dụng để đặt nhãn cho hệ thống tệp. Nhãn này có thể có tối đa 16 ký tự. Tham số `-U` sẽ đặt UUID (số thập lục phân 128 bit) cho hệ thống tệp. Trong ví dụ trên, UUID của hệ thống tệp là `6e2c12e3-472d-4bac-a257-c49ac07f3761`. Cả nhãn và UUID đều có thể được sử dụng thay cho tên thiết bị (như `/dev/sda1`) để gắn kết hệ thống tệp.

Tùy chọn `-e BEHAVIOUR` sẽ xác định hành vi của hạt nhân khi tìm thấy lỗi hệ thống tệp. Có ba hành vi có thể xảy ra:

`continue`
: Sẽ tiếp tục thực thi bình thường.

`remount-ro`
: Sẽ gắn kết lại hệ thống tệp dưới dạng chỉ đọc.

`panic`
: Sẽ gây ra lỗi rối loạn hạt nhân.

Hành vi mặc định sẽ là `continue` (tiếp tục). `remount-ro` có thể sẽ hữu ích trong các ứng dụng nhạy cảm với dữ liệu vì nó sẽ ngay lập tức ngừng ghi vào đĩa để tránh nhiều lỗi tiềm ẩn.

Hệ thống tệp ext3 về cơ bản là hệ thống tệp ext2 nhưng có thêm nhật ký. Khi sử dụng `tune2fs`, ta có thể thêm nhật ký vào hệ thống tệp ext2 và từ đó chuyển đổi nó thành ext3. Quy trình này rất đơn giản, ta chỉ cần truyền tham số `-j` cho `tune2fs`, theo sau là thiết bị chứa hệ thống tệp:

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

Sau đó, khi gắn kết hệ thống tệp đã chuyển đổi, đừng quên đặt loại thành `ext3` để có thể sử dụng nhật ký.

Khi xử lý các hệ thống tệp được ghi nhật ký, tham số `-J` cho phép ta sử dụng các tham số bổ sung

để đặt một số tùy chọn nhật ký như `-J size=` (để đặt kích thước nhật ký, tính bằng megabyte), `-J location=` (để chỉ định vị trí nhật ký phải được lưu trữ, có thể là một khối cụ thể hoặc một vị trí cụ thể trên đĩa có hậu tố như `M` hoặc `G`) và `-J device=` (để đặt nhật ký trên một thiết bị ngoại vi).

Ta có thể chỉ định nhiều tham số cùng một lúc bằng cách phân tách chúng bằng dấu phẩy. Ví dụ: `-J size=10,location=100M,device=/dev/sdb1` sẽ tạo Nhật ký 10 MB ở vị trí 100 MB trên thiết bị `/dev/sdb1`.

WARNING

`tune2fs` có một tùy chọn "quyền năng" là `-f`. Tùy chọn này sẽ buộc nó hoàn thành một thao tác ngay cả khi tìm thấy lỗi. Vì là một tuỳ chọn rất mạnh nên nó phải được sử dụng một cách hết sức thận trọng.

Duy trì Hệ thống Tệp XFS

Đối với hệ thống tệp XFS, `xfs_repair` chính là công cụ tương đương với `fsck`. Nếu còn nghi ngờ có điều gì đó không ổn với hệ thống tệp, điều đầu tiên chúng ta cần làm là quét hệ thống tệp để tìm ra lỗi.

Tác vụ này có thể được thực hiện bằng cách truyền tham số `-n` cho `xfs_repair`, theo sau là thiết bị chứa hệ thống tệp. Tham số `-n` có nghĩa là "no modify" (không sửa đổi): hệ thống tệp sẽ được kiểm tra, lỗi sẽ được báo cáo nhưng sẽ không có sửa chữa nào được thực hiện:

```
# 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.
```

Nếu tìm thấy lỗi, ta có thể tiến hành sửa chữa mà không cần tham số `-n` như sau: `xfs_repair /dev/sdb1`.

`xfs_repair` có chấp nhận một số tùy chọn dòng lệnh. Trong số đó:

`-l LOGDEV` và `-r RTDEV`

Những tuỳ chọn này sẽ cần thiết nếu hệ thống tệp có các phần nhật ký và thời gian thực bên ngoài. Trong trường hợp này, hãy thay thế `LOGDEV` và `RTDEV` bằng các thiết bị tương ứng.

`-m N`

Được sử dụng để giới hạn mức sử dụng bộ nhớ của `xfs_repair` ở mức `N` megabyte, điều này có thể sẽ hữu ích trong cài đặt máy chủ. Theo như trang hướng dẫn, `xfs_repair` sẽ chia tỷ lệ sử dụng bộ nhớ khi cần theo mặc định. Nó có thể lên tới 75% RAM vật lý của hệ thống.

`-d`

Chế độ "nguy hiểm" (dangerous) sẽ cho phép ta sửa chữa các hệ thống tệp được gắn kết ở chế độ chỉ đọc.

`-v`

Rất dễ đoán: đây chính là chế độ chi tiết. Mỗi lần tham số này được sử dụng, "mức độ chi tiết" sẽ được tăng lên (ví dụ: `-v -v` sẽ in nhiều thông tin hơn so với `-v`).

Hãy lưu ý rằng `xfs_repair` không thể sửa chữa hệ thống tệp có nhật ký "xấu". Ta có thể "xóa bỏ" một nhật ký xấu với tham số `-L`, nhưng hãy nhớ rằng đây là một *hạ sách* vì nó có thể dẫn đến hỏng hệ thống tệp và mất dữ liệu.

Để gỡ lỗi hệ thống tệp XFS, ta có thể sử dụng tiện ích `xfs_db` (như trong `xfs_db /dev/sdb1`). Lệnh này chủ yếu được sử dụng để kiểm tra các thành phần và thông số khác nhau của hệ thống tệp.

Tiện ích này có lời nhắc tương tác (giống như `parted`) với nhiều lệnh nội bộ. Một hệ thống trợ giúp cũng có sẵn: hãy gõ `help` để xem danh sách tất cả các lệnh và `help` theo sau là tên lệnh để xem thêm thông tin về lệnh.

Một tiện ích hữu ích khác là `xfs_fsr` có thể được sử dụng để sắp xếp lại ("chống phân mảnh") một hệ thống tệp XFS. Khi được thực thi mà không có bất kỳ đối số bổ sung nào, nó sẽ chạy trong hai giờ và cố gắng chống phân mảnh tất cả các hệ thống tệp XFS được gắn kết và có thể ghi được liệt kê trong tệp `/etc/mtab/`. Chúng ta có thể sẽ cần cài đặt tiện ích này bằng trình quản lý gói dành cho bản phân phối Linux của mình vì nó có thể không phải là một phần của bản cài đặt mặc định. Để biết thêm thông tin, hãy tham khảo trang hướng dẫn tương ứng.

Bài tập Hướng dẫn

1. Bằng cách sử dụng `du`, làm cách nào để có thể kiểm tra xem các tệp trong thư mục hiện tại đang sử dụng bao nhiêu dung lượng?

2. Bằng cách sử dụng `df`, hãy liệt kê thông tin cho mọi hệ thống tệp ext4 với đầu ra bao gồm các trường và theo thứ tự sau: thiết bị, điểm gắn kết, tổng số nút định danh, số nút định danh có sẵn, phần trăm dung lượng trống.

3. Lệnh để chạy `e2fsck` trên `/dev/sdc1` ở chế độ không tương tác trong khi vẫn cố tự động sửa hầu hết các lỗi là gì?

4. Giả sử `/dev/sdb1` là một hệ thống tệp ext2. Làm cách để có thể chuyển đổi nó thành ext3, đồng thời đặt lại số lần gắn kết của nó và thay đổi nhãn của nó thành `UserData`?

5. Làm cách nào để có thể kiểm tra lỗi trên hệ thống tệp XFS mà không sửa chữa bất kỳ lỗi nào được tìm thấy?

Bài tập Mở rộng

1. Giả sử bạn có hệ thống tệp ext4 trên `/dev/sda1` với các tham số sau thu được bằng `tune2fs`:

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

   Điều gì sẽ xảy ra ở lần khởi động tiếp theo nếu lệnh `tune2fs -c 9 /dev/sda1` được gọi?

2. Hãy xem xét đầu ra sau của `du -h`:

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

   Có bao nhiêu dung lượng bị chiếm bởi các tệp trong thư mục hiện tại? Làm cách nào để có thể viết lại lệnh để hiển thị thông tin này rõ ràng hơn?

3. Điều gì sẽ xảy ra với hệ thống tệp ext2 `/dev/sdb1` nếu lệnh bên dưới được gọi?

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

4. Làm cách nào để có thể kiểm tra lỗi trên hệ thống tệp XFS ở `/dev/sda1` có phần nhật ký trên `/dev/sdc1` mà không thực sự thực hiện bất kỳ sửa chữa nào?

5. Sự khác biệt giữa tham số `-T` và `-t` cho lệnh `df` là gì?

Tóm tắt

Trong bài học này, chúng ta đã học về:

- Cách kiểm tra dung lượng đã sử dụng và dung lượng trống trên hệ thống tệp.
- Cách điều chỉnh đầu ra của `df` cho phù hợp với nhu cầu của người dùng.
- Cách kiểm tra tính toàn vẹn và sửa chữa hệ thống tệp bằng `fsck` và `e2fsck`.
- Cách tinh chỉnh hệ thống tệp ext bằng `tune2fs`.
- Cách kiểm tra và sửa chữa hệ thống tệp XFS bằng `xfs_repair`.

Các lệnh sau đã được thảo luận trong bài học này:

`du`
: Xem dung lượng đĩa đang được sử dụng trên hệ thống tệp.

`df`
: Xem dung lượng đĩa có sẵn trên hệ thống tệp.

`fsck`
: Tiện ích sửa chữa kiểm tra hệ thống tệp.

`e2fsck`
: Tiện ích sửa chữa kiểm tra hệ thống tệp dành riêng cho các hệ thống tệp ext (ext2/3/4).

`tune2fs`
: Sửa đổi các tham số hệ thống tệp trên hệ thống tệp mở rộng (ext2/3/4).

`xfs_repair`
: Tương đương với `fsck` cho các hệ thống tệp XFS.

`xfs_db`
: Tiện ích này được sử dụng để xem các tham số khác nhau của hệ thống tệp XFS.

Đáp án Bài tập Hướng dẫn

1. Bằng cách sử dụng `du`, làm cách nào để có thể kiểm tra xem các tệp trong thư mục hiện tại đang sử dụng bao nhiêu dung lượng?

   Đầu tiên, sử dụng tham số `-S` để tách đầu ra của thư mục hiện tại khỏi các thư mục con của nó. Sau đó, sử dụng `-d 0` để giới hạn độ sâu đầu ra về 0, nghĩa là "không thư mục con". Đừng quên tham số `-h` để nhận đầu ra ở định dạng "con người có thể đọc được`":

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

   hoặc

   ```
   $ du -Shd 0
   ```

2. Bằng cách sử dụng `df`, hãy liệt kê thông tin cho mọi hệ thống tệp ext4 với đầu ra bao gồm các trường và theo thứ tự sau: thiết bị, điểm gắn kết, tổng số nút định danh, số nút định danh có sẵn, phần trăm dung lượng trống.

   Bạn có thể lọc các hệ thống tệp bằng tùy chọn `-t`, theo sau là tên hệ thống tệp. Để có được đầu ra cần thiết, hãy sử dụng `--output=source,target,itotal,iavail,pcent`. Vì vậy, câu trả lời sẽ là:

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

3. Lệnh để chạy `e2fsck` trên `/dev/sdc1` ở chế độ không tương tác trong khi vẫn cố tự động sửa hầu hết các lỗi là gì?

   Tham số để tự động sửa hầu hết các lỗi là `-p`. Vì vậy, câu trả lời sẽ là:

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

4. Giả sử `/dev/sdb1` là một hệ thống tệp ext2. Làm cách để có thể chuyển đổi nó thành ext3, đồng thời đặt lại số lần gắn kết của nó và thay đổi nhãn của nó thành `UserData`?

   Hãy nhớ rằng việc chuyển đổi một hệ thống tệp ext2 thành ext3 chỉ đơn thuần là ghi thêm nhật ký. Việc này có thể được thực hiện với tham số `-j`. Để đặt lại số lần gắn kết, hãy sử dụng `-C 0`. Để thay đổi nhãn, hãy sử dụng `-L UserData`. Câu trả lời đúng sẽ là:

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

5. Làm cách nào để có thể kiểm tra lỗi trên hệ thống tệp XFS mà không sửa chữa bất kỳ lỗi nào được tìm thấy?

   Sử dụng tham số `-n` (giống như trong `xfs -n`), theo sau là thiết bị tương ứng.

Đáp án Bài tập Mở rộng

1. Giả sử bạn có hệ thống tệp ext4 trên `/dev/sda1` với các tham số sau thu được bằng `tune2fs`:

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

   Điều gì sẽ xảy ra ở lần khởi động tiếp theo nếu lệnh `tune2fs -c 9 /dev/sda1` được gọi?

   Lệnh sẽ đặt số lần gắn kết tối đa cho hệ thống tệp là 9. Vì số lần gắn kết hiện tại là 8 nên lần khởi động hệ thống tiếp theo sẽ tiến hành kiểm tra hệ thống tệp.

2. Hãy xem xét đầu ra sau của `du -h`:

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

   Có bao nhiêu dung lượng bị chiếm bởi các tệp trong thư mục hiện tại? Làm cách nào để có thể viết lại lệnh để hiển thị thông tin này rõ ràng hơn?

   Trong tổng số 232K đã sử dụng, có 224K được sử dụng bởi thư mục con `somedir` và các thư mục con của nó. Vì vậy, khi loại trừ những thư mục đó, chúng ta còn 8K được sử dụng bởi các tệp trong thư mục hiện tại. Thông tin này có thể được hiển thị rõ ràng hơn bằng cách sử dụng tham số `-S`. Tham số này sẽ giúp phân tách thư mục hiện tại và thư mục somedir để xác định dung lượng của các tệp trong thư mục hiện tại (trong lượt kiểm tra này)

3. Điều gì sẽ xảy ra với hệ thống tệp ext2 `/dev/sdb1` nếu lệnh bên dưới được gọi?

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

   Một nhật ký sẽ được thêm vào `/dev/sdb1` để chuyển đổi nó thành ext3. Nhật ký sẽ được lưu trữ trên thiết bị `/dev/sdc1` và hệ thống tệp sẽ được kiểm tra sau mỗi 30 ngày.

4. Làm cách nào để có thể kiểm tra lỗi trên hệ thống tệp XFS ở `/dev/sda1` có phần nhật ký trên `/dev/sdc1` mà không thực sự thực hiện bất kỳ sửa chữa nào?

   Sử dụng `xfs_repair`, theo sau là `-l /dev/sdc1` để biểu thị thiết bị chứa phần nhật ký và `-n` để tránh thực hiện bất kỳ một thay đổi nào.

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

5. Sự khác biệt giữa tham số `-T` và `-t` cho lệnh `df` là gì?

   Tham số `-T` sẽ bao gồm cả loại của từng hệ thống tệp trong đầu ra của `df`. `-t` là một bộ lọc và sẽ chỉ hiển thị các hệ thống tệp thuộc một loại nhất định ở đầu ra và loại trừ tất cả các hệ thống khác.

104.3 Kiểm soát việc gắn và ngắt gắn kết Hệ thống Tệp

Tham khảo các mục tiêu LPI

LPIC-1 version 5.0, Exam 101, Objective 104.3

Khối lượng

3

Các lĩnh vực kiến thức chính

- Gắn và ngắt gắn kết hệ thống tệp theo cách thủ công.
- Định cấu hình gắn hệ thống tệp khi khởi động.
- Định cấu hình hệ thống tệp di động có thể gắn kết của người dùng
- Sử dụng nhãn và UUID để xác định và gắn hệ thống tệp.
- Nhận thức về các đơn vị gắn kết systemd.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

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

104.3 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	104 Thiết bị, Hệ thống tệp Linux, Tiêu chuẩn phân cấp hệ thống Tệp
Mục tiêu:	104.3 Kiểm soát việc gắn và ngắt gắn kết Hệ thống Tệp
Bài:	1 trên 1

Giới thiệu

Cho đến nay, chúng ta đã được học về cách phân vùng đĩa cũng như cách tạo và duy trì hệ thống tệp trên đó. Tuy nhiên, trước khi có thể truy cập được hệ thống tệp trên Linux, nó cần được *gắn kết* (mount).

Điều này có nghĩa là gắn kết hệ thống tệp vào một điểm cụ thể trong cây thư mục của hệ thống được gọi là *điểm gắn kết*. Hệ thống tệp có thể được gắn kết thủ công hoặc tự động. Trong bài học này, chúng ta sẽ cùng tìm hiểu về một trong số nhiều cách để thực hiện tác vụ nói trên.

Gắn kết và Ngắt Gắn kết Hệ thống Tệp

Lệnh gắn kết hệ thống tệp theo cách thủ công được gọi là `mount` và cú pháp của nó là:

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

Trong đó:

`TYPE`

Loại hệ thống tệp được gắn kết (ví dụ: ext4, btrfs, exfat, v.v.).

`DEVICE`

Tên của phân vùng chứa hệ thống tệp (ví dụ: `/dev/sdb1`)

`MOUNTPOINT`

Nơi hệ thống tệp sẽ được gắn kết. Thư mục được gắn kết không nhất thiết phải trống nhưng bắt buộc phải tồn tại. Tuy nhiên, các tệp trong thư mục đó sẽ không thể được truy cập theo tên trong khi hệ thống tệp được gắn kết.

Ví dụ: để gắn ổ flash USB chứa hệ thống tệp exFAT nằm trên `/dev/sdb1` vào thư mục có tên `flash` trong thư mục chính, ta có thể sử dụng:

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

TIP

Có nhiều hệ thống Linux sử dụng vỏ Bash và trên các hệ thống đó, dấu ngã `~` trên đường dẫn đến điểm gắn kết chính là cách viết tắt của thư mục chính của người dùng hiện tại. Ví dụ: nếu người dùng hiện tại được đặt tên là `john`, nó sẽ được thay thế bằng `/home/john`.

Sau khi gắn kết, nội dung của hệ thống tệp sẽ có thể truy cập được trong thư mục `~/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
```

Liệt kê các Hệ thống Tệp được gắn kết

Nếu chỉ nhập `mount`, chúng ta sẽ nhận được danh sách tất cả các hệ thống tệp hiện được gắn kết trên hệ thống. Danh sách này có thể sẽ khá dài vì ngoài các đĩa được gắn kết vào hệ thống thì nó còn chứa một số hệ thống tệp thời gian chạy trong bộ nhớ để phục vụ nhiều mục đích khác nhau. Để lọc đầu ra, ta có thể sử dụng tham số `-t` để chỉ liệt kê các hệ thống tệp thuộc loại tương ứng như dưới đây:

```
# mount -t ext4
```

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

Chúng ta có thể chỉ định nhiều hệ thống tệp cùng một lúc bằng cách phân tách chúng bằng dấu phẩy:

```
# 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]
```

Đầu ra trong các ví dụ trên có thể được mô tả theo định dạng:

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

Trong đó, `SOURCE` là phân vùng chứa hệ thống tệp, `TARGET` là thư mục nơi nó được gắn kết, `TYPE` là loại hệ thống tệp và `OPTIONS` là các tùy chọn được truyền cho lệnh `mount` tại thời điểm gắn kết.

Tham số Dòng lệnh bổ sung

Có nhiều tham số dòng lệnh có thể được sử dụng với `mount`. Một trong các tham số được sử dụng nhiều nhất là:

`-a`

Tham số này sẽ gắn kết tất cả các hệ thống tệp được liệt kê trong tệp `/etc/fstab` (xem thêm trong phần tiếp theo).

`-o` hoặc `--options`

Tham số này sẽ truyền một danh sách *các tuỳ chọn gắn kết* được phân tách bằng dấu phẩy sang lệnh mount và có thể thay đổi cách hệ thống tệp được gắn kết. Các tuỳ chọn này cũng sẽ được thảo luận cùng với `/etc/fstab`.

`-r` hoặc `-ro`

Tuỳ chọn này sẽ gắn kết hệ thống tệp ở dạng chỉ đọc.

`-w` hoặc `-rw`

Tuỳ chọn này sẽ gắn hệ thống tệp ở dạng có thể ghi.

Để ngắt gắn kết hệ thống tệp, hãy sử dụng lệnh `umount`, theo sau là tên thiết bị hoặc điểm gắn kết. Trong ví dụ trên, các lệnh bên dưới đây có thể được hoán đổi cho nhau:

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

Một số tham số dòng lệnh cho `umount` là:

`-a`

Tham số này sẽ ngắt gắn kết tất cả các hệ thống tệp được liệt kê trong `/etc/fstab`.

`-f`

Tham số này sẽ buộc ngắt gắn kết một hệ thống tệp. Nó có thể sẽ hữu ích khi chúng ta gắn kết một hệ thống tệp từ xa hiện không thể truy cập được.

`-r`

Nếu hệ thống tệp không thể ngắt gắn kết được, tham số này sẽ cố gắng biến nó thành chế độ chỉ đọc.

Xử lý các Tệp đang mở

Khi ngắt gắn kết hệ thống tệp, chúng ta có thể sẽ gặp thông báo lỗi cho biết mục tiêu đang bận (`target is busy`). Điều này sẽ xảy ra nếu bất kỳ tệp nào trên hệ thống tệp đang mở. Tuy nhiên, chúng ta khó có thể ngay lập tức biết được vị trí của tệp đang mở hoặc cái gì đang truy cập vào hệ thống tệp.

Trong những trường hợp như vậy, ta có thể sử dụng lệnh `lsof`, theo sau là tên của thiết bị chứa hệ thống tệp để xem danh sách các tiến trình đang truy cập nó và tệp nào đang mở. Ví dụ:

```
# 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` là tên của lệnh thực thi đã mở tệp, `PID` là số tiến trình và `NAME` là tên của tệp đang mở. Trong ví dụ trên, tệp `MagPi40.pdf` được mở bởi chương trình `evince` (trình xem PDF). Nếu đóng chương trình, chúng ta sẽ có thể ngắt gắn kết hệ thống tệp.

NOTE Trước khi đầu ra `lsof` xuất hiện, người dùng trong môi trường `GNOME` sẽ thấy thông báo cảnh báo trong cửa sổ dòng lệnh.

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

`lsof` sẽ cố gắng xử lý tất cả các hệ thống tệp được gắn kết. Thông báo cảnh báo này được đưa ra vì `lsof` đã gặp phải hệ thống tệp ảo `GNOME` (GVFS). Đây là trường hợp đặc biệt của hệ thống tệp trong không gian người dùng (`FUSE`). Nó hoạt động như một cầu nối giữa `GNOME`, các API của nó và hạt nhân. Không có ai—kể cả siêu người dùng—có thể truy cập vào một trong các hệ thống tệp này ngoại trừ chủ sở hữu đã gắn kết nó (trong trường hợp này là `GNOME`). Chúng ta có thể bỏ qua cảnh báo này.

Nên gắn kết ở đâu?

Ta có thể gắn kết hệ thống tệp vào bất cứ đâu mà mình muốn. Tuy nhiên, có một số quy tắc hữu ích cần được tuân theo để giúp việc quản trị hệ thống trở nên dễ dàng hơn.

Theo truyền thống, `/mnt` là thư mục mà tất cả các thiết bị ngoại vi sẽ được gắn kết vào và một số "điểm neo" được cấu hình sẵn cho các thiết bị phổ biến như ổ đĩa quang CD-ROM (`/mnt/cdrom`) và đĩa mềm (`/mnt /floppy`) sẽ tồn tại dưới nó.

Thư mục này đã được thay thế bởi `/media` - hiện là điểm gắn kết mặc định cho bất kỳ phương tiện nào mà người dùng có thể tháo rời (ví dụ: ổ đĩa ngoại vi, ổ USB flash, đầu đọc thẻ nhớ, v.v.) được kết nối với hệ thống.

Trên hầu hết các bản phân phối Linux hiện đại và môi trường máy tính để bàn, các thiết bị di động sẽ được tự động gắn kết vào `/media/USER/LABEL` khi được kết nối với hệ thống. Trong đó, `USER` là tên người dùng và `LABEL` là nhãn thiết bị. Ví dụ: một ổ flash USB có nhãn `FlashDrive` được gắn kết bởi người dùng `john` sẽ được gắn vào `/media/john/FlashDrive/`. Cách xử lý việc này sẽ tùy thuộc vào từng môi trường máy tính để bàn.

Tuy là vậy, bất cứ khi nào chúng ta cần gắn kết một hệ thống tệp theo *cách thủ công*, cách tốt nhất là gắn kết nó vào `/mnt`.

Gắn kết Hệ thống Tệp khi khởi động

Tệp `/etc/fstab` sẽ chứa các mô tả về hệ thống tệp có thể được gắn kết. Đây là một tệp văn bản mà ở trong đó, mỗi dòng sẽ mô tả một hệ thống tệp được gắn kết với sáu trường theo thứ tự sau:

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

Trong đó:

`FILESYSTEM`

Thiết bị chứa hệ thống tệp sẽ được gắn kết. Thay vì thiết bị, ta có thể chỉ định UUID hoặc nhãn của phân vùng (chúng ta sẽ thảo luận sau).

`MOUNTPOINT`

Nơi hệ thống tệp sẽ được gắn kết.

`TYPE`

Loại hệ thống tệp.

`OPTIONS`

Các tùy chọn gắn kết sẽ được truyền cho `mount`.

`DUMP`

Cho biết liệu có nên xem xét bất kỳ hệ thống tệp ext2, ext3 hoặc ext4 nào để sao lưu bằng lệnh `dump` hay không. Thông thường nó sẽ bằng 0, nghĩa là chúng nên được bỏ qua.

`PASS`

Khi khác 0, nó sẽ xác định thứ tự các hệ thống tệp sẽ được kiểm tra khi khởi động. Thông thường nó sẽ bằng 0.

Ví dụ: phân vùng đầu tiên trên đĩa đầu tiên của máy có thể được mô tả là:

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

Các tùy chọn gắn kết trên `OPTIONS` là một danh sách các tham số được phân tách bằng dấu phẩy. Chúng có thể là tham số chung hoặc dành riêng cho hệ thống tệp. Trong số những tham số chung, chúng ta có:

`atime` và `noatime`

Theo mặc định, mỗi khi tệp được đọc thì thông tin về thời gian truy cập sẽ được cập nhật. Việc tắt tính năng này (với `noatime`) có thể tăng tốc độ I/O của đĩa. Đừng nhầm lẫn điều này với thời gian sửa đổi được cập nhật mỗi khi tệp được ghi vào.

`auto` và `noauto`

Khiến hệ thống tệp có thể (hoặc không thể) được gắn kết tự động với `mount -a`.

`defaults`

Tuỳ chọn này sẽ truyền các tùy chọn `rw`, `suid`, `dev`, `exec`, `auto`, `nouser` và `async` sang cho `mount`.

`dev` và `nodev`

Quyết định xem thiết bị dạng ký tự hoặc thiết bị khối trong hệ thống tệp được gắn kết có nên được diễn giải hay không.

`exec` và `noexec`

Cho phép hoặc từ chối quyền thực thi các tệp nhị phân trên hệ thống tệp.

`user` và `nouser`

Cho phép (hoặc không cho phép) người dùng bình thường gắn kết hệ thống tệp.

`group`

Cho phép người dùng gắn kết hệ thống tệp nếu người dùng thuộc cùng nhóm sở hữu thiết bị chứa nó.

`owner`

Cho phép người dùng gắn kết hệ thống tệp nếu người dùng sở hữu thiết bị chứa nó.

`suid` và `nosuid`

Cho phép hoặc không cho phép các bit SETUID và SETGID có hiệu lực.

`ro` và `rw`

Gắn kết hệ thống tệp ở dạng chỉ đọc hoặc có thể ghi.

`remount`

Tuỳ chọn này sẽ cố gắng gắn kết lại một hệ thống tệp đã được gắn kết. Tham số này không được sử dụng trên `/etc/fstab` mà là tham số cho `mount -o`. Ví dụ: để gắn kết lại phân vùng `/dev/sdb1` đã được gắn ở chế độ chỉ đọc, ta có thể sử dụng lệnh `mount -o remount,ro /dev/sdb1`. Khi gắn kết lại, chúng ta không cần chỉ định loại hệ thống tệp mà chỉ cần có tên thiết bị *hoặc* điểm gắn kết.

`sync` và `async`

Thực hiện tất cả thao tác I/O với hệ thống tệp một cách đồng bộ (hoặc không đồng bộ). `async` (không đồng bộ) thường là mặc định. Trang hướng dẫn của `mount` có cảnh báo rằng việc sử dụng `sync` trên phương tiện có số chu kỳ ghi hạn chế (như ổ đĩa flash hoặc thẻ nhớ) có thể rút ngắn tuổi thọ của thiết bị.

Sử dụng UUID và Nhãn

Việc chỉ định tên của thiết bị chứa hệ thống tệp cần gắn kết có thể gây ra một số vấn đề. Đôi khi, cùng một tên thiết bị có thể được gán cho một thiết bị khác, tùy thuộc vào thời điểm hoặc vị trí

thiết bị đó được kết nối với hệ thống. Ví dụ: ổ flash USB trên `/dev/sdb1` có thể được gán cho `/dev/sdc1` nếu được cắm trên một cổng khác hoặc sau một ổ flash khác.

Một cách để tránh điều này là chỉ định nhãn hoặc UUID (*Mã định danh duy nhất toàn cầu*) của ổ phân vùng. Cả hai đều được chỉ định khi hệ thống tệp được tạo và sẽ không thay đổi trừ khi hệ thống tệp bị hủy hoặc được gán nhãn hoặc UUID mới theo cách thủ công.

Lệnh `lsblk` có thể được sử dụng để truy vấn thông tin về hệ thống tệp và tìm ra nhãn và UUID được liên kết với nó. Để thực hiện việc này, hãy sử dụng tham số `-f`, theo sau là tên thiết bị:

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

Dưới đây là ý nghĩa của từng cột:

`NAME`
: Tên của thiết bị chứa hệ thống tệp.

`FSTYPE`
: Loại hệ thống tệp.

`LABEL`
: Nhãn hệ thống tệp.

`UUID`
: Mã định danh duy nhất toàn cầu (UUID) được gán cho hệ thống tệp.

`FSAVAIL`
: Dung lượng có sẵn trong hệ thống tệp.

`FSUSE%`
: Phần trăm sử dụng của hệ thống tệp.

`MOUNTPOINT`
: Nơi hệ thống tệp sẽ được gắn kết.

Trong `/etc/fstab`, một thiết bị có thể được chỉ định bởi UUID của nó với tùy chọn `UUID=`, theo sau là UUID hoặc với `LABEL=`, theo sau là nhãn. Vì vậy, thay vì:

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

Chúng ta sẽ sử dụng:

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

Hoặc, nếu ta có một đĩa có nhãn `homedisk`:

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

Cú pháp tương tự có thể được sử dụng với lệnh `mount`. Thay vì tên thiết bị, hãy truyền UUID hoặc nhãn. Ví dụ: để gắn một đĩa NTFS ngoại vi với UUID `56C11DCC5D2E1334` trên `/mnt/external`, lệnh sẽ là:

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

Gắn kết Đĩa với Systemd

Systemd là tiến trình init - tiến trình đầu tiên chạy trên nhiều bản phân phối Linux. Nó chịu trách nhiệm tạo ra các tiến trình khác, khởi động các dịch vụ và khởi động hệ thống. Trong số nhiều tác vụ khác, systemd cũng có thể được sử dụng để quản lý việc gắn kết (và tự động gắn kết) các hệ thống tệp.

Để sử dụng tính năng này của systemd, chúng ta cần tạo một tệp cấu hình được gọi là *đơn vị gắn kết*. Mỗi ổ phân vùng cần gắn kết sẽ có đơn vị gắn riêng và chúng cần được đặt trong `/etc/systemd/system/`.

Đơn vị gắn kết là các tệp văn bản đơn giản có phần mở rộng `.mount`. Định dạng cơ bản sẽ như dưới đây:

```
[Unit]
Description=

[Mount]
What=
Where=
Type=
Options=
```

```
[Install]
WantedBy=
```

`Description=`

Mô tả ngắn gọn về đơn vị gắn kết, đại loại như `Mounts the backup disk`.

`What=`

Cái gì nên được gắn kết. Ổ phân vùng phải được chỉ định là `/dev/disk/by-uuid/VOL_UUID`, trong đó, `VOL_UUID` là UUID của ổ phân vùng.

`Where=`

Đường dẫn đầy đủ đến nơi ổ phân vùng sẽ được gắn vào.

`TYPE`

Loại hệ thống tệp.

`Options=`

Các tùy chọn gắn kết mà ta có thể muốn truyền qua. Những tùy chọn này khi được sử dụng với lệnh `mount` hoặc trong `/etc/fstab` sẽ giống như nhau.

`WantedBy=`

Được sử dụng để quản lý các phần phụ thuộc. Trong trường hợp này, chúng ta sẽ sử dụng `multi-user.target`, có nghĩa là bất cứ khi nào hệ thống khởi động vào môi trường đa người dùng (khởi động bình thường), đơn vị sẽ được gắn kết.

Ví dụ trước của chúng ta về đĩa ngoại vi có thể được viết như sau:

```
[Unit]
Description=External data disk

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

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

Nhưng vẫn chưa xong. Để có thể hoạt động một cách chuẩn xác, đơn vị gắn kết *phải* có cùng tên

với điểm gắn kết. Trong trường hợp này, điểm gắn kết là `/mnt/external`. Vì vậy, tệp phải được đặt tên là `mnt-external.mount`.

Sau đó, ta cần khởi động lại trình nền systemd bằng lệnh `systemctl` và khởi động đơn vị:

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

Bây giờ, nội dung của đĩa ngoại vi sẽ có sẵn trên `/mnt/external`. Ta có thể kiểm tra trạng thái của quá trình gắn kết bằng lệnh `systemctl status mnt-external.mount` như bên dưới đây:

```
# 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.
```

Lệnh `systemctl start mnt-external.mount` sẽ chỉ kích hoạt thiết bị cho phiên hiện tại. Nếu muốn bật nó trong mỗi lần khởi động, hãy thay thế `start` bằng `enable`:

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

Tự động gắn một Đơn vị Gắn kết

Các đơn vị gắn kết có thể được tự động gắn kết bất cứ khi nào điểm gắn kết được truy cập. Để thực hiện việc này, ta cần có tệp `.automount` cùng với tệp `.mount` mô tả đơn vị. Định dạng cơ bản sẽ là:

```
[Unit]
Description=

[Automount]
Where=
```

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

Giống như trước đây, `Description=` là một mô tả ngắn về tệp, và `Where=` là điểm gắn kết. Ví dụ: tệp `.automount` trong ví dụ trước sẽ là:

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

[Automount]
Where=/mnt/external

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

Lưu tệp có cùng tên với điểm gắn kết (trong trường hợp này là `mnt-external.automount`), tải lại systemd và khởi động đơn vị:

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

Bây giờ, bất cứ khi nào thư mục `/mnt/external` được truy cập, đĩa sẽ được gắn kết. Giống như trước đây, để bật tính năng tự động gắn trong mỗi lần khởi động, chúng ta sẽ sử dụng:

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

Bài tập Hướng dẫn

1. Bằng cách sử dụng `mount` với tuỳ chọn `noatime` và `async`, làm cách nào để có thể gắn kết hệ thống tệp `ext4` trên `/dev/sdc1` vào `/mnt/external` dưới dạng chỉ đọc?

2. Khi ngắt gắn kết hệ thống tệp tại `/dev/sdd2`, bạn nhận được thông báo lỗi `target is busy`. Làm cách nào để có thể biết tệp nào trên hệ thống tệp đang mở và tiến trình nào đã mở chúng?

3. Hãy xem xét mục sau trong `/etc/fstab`: `/dev/sdb1 /data ext4 noatime,noauto,async`. Nếu lệnh `mount -a` được đưa ra, hệ thống tệp này có được gắn kết không? Tại sao?

4. Làm cách nào để có thể tìm ra UUID của hệ thống tệp trong `/dev/sdb1`?

5. Làm cách nào để sử dụng `mount` để gắn kết lại hệ thống tệp exFAT có UUID là `6e2c12e3-472d-4bac-a257-c49ac07f3761` được gắn tại `/mnt/data` dưới dạng chỉ đọc?

6. Làm cách nào để có được danh sách tất cả các hệ thống tệp `ext3` và `ntfs` hiện được gắn trên một hệ thống?

Bài tập Mở rộng

1. Hãy xem xét mục sau trong `/etc/fstab`: `/dev/sdc1 /backup ext4 noatime,nouser,async`. Người dùng có thể gắn kết hệ thống tệp này bằng lệnh `mount /backup` không? Tại sao?

2. Giả sử có một hệ thống tệp từ xa được gắn kết tại `/mnt/server`, hệ thống này hiện không thể truy cập được do mất kết nối mạng. Làm cách nào để có thể buộc nó phải ngắt gắn kết, hoặc nếu không thể thì được gắn ở dạng chỉ đọc?

3. Hãy viết một mục nhập `/etc/fstab` sẽ gắn kết một ổ phân vùng btrfs với nhãn `Backup` trên `/mnt/backup` với các tùy chọn mặc định và không cho phép thực thi các tệp nhị phân từ nó.

4. Hãy xem đơn vị gắn kết systemd sau:

```
[Unit]
Description=External data disk

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

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

   - Mục nhập `/etc/fstab` tương đương cho hệ thống tệp này là gì?

5. Tên tệp của đơn vị ở trên phải là gì để systemd có thể sử dụng nó? Nó nên được đặt ở đâu?

Tóm tắt

Trong bài học này, chúng ta đã học cách gắn kết và ngắt gắn kết hệ thống tệp theo cách thủ công hoặc tự động. Một số lệnh và khái niệm đã được giải thích là:

- `mount` (gắn kết thiết bị vào một vị trí)
- `umount` (ngắt gắn kết thiết bị)
- `lsof` (liệt kê các tiến trình truy cập hệ thống tệp)
- thư mục `/mnt` và `/media`
- `/etc/fstab`
- `lsblk` (liệt kê loại và UUID của hệ thống tệp)
- Cách gắn kết hệ thống tệp bằng UUID hoặc nhãn của nó.
- Cách gắn kết hệ thống tệp bằng cách sử dụng các đơn vị gắn kết systemd.
- Cách tự động gắn kết hệ thống tệp bằng cách sử dụng các đơn vị gắn kết systemd.

Đáp án Bài tập Hướng dẫn

1. Bằng cách sử dụng `mount` với tuỳ chọn `noatime` và `async`, làm cách nào để có thể gắn kết hệ thống tệp `ext4` trên `/dev/sdc1` vào `/mnt/external` dưới dạng chỉ đọc?

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

2. Khi ngắt gắn kết hệ thống tệp tại `/dev/sdd2`, bạn nhận được thông báo lỗi `target is busy`. Làm cách nào để có thể biết tệp nào trên hệ thống tệp đang mở và tiến trình nào đã mở chúng?

   Sử dụng `lsof`, theo sau là tên thiết bị:

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

3. Hãy xem xét mục sau trong `/etc/fstab`: `/dev/sdb1 /data ext4 noatime,noauto,async`. Nếu lệnh `mount -a` được đưa ra, hệ thống tệp này có được gắn kết không? Tại sao?

   Nó sẽ không được gắn kết. Mấu chốt ở đây là tham số `noauto`, có nghĩa là mục này sẽ bị `mount -a` bỏ qua.

4. Làm cách nào để có thể tìm ra UUID của hệ thống tệp trong `/dev/sdb1`?

   Sử dụng `lsblk -f`, theo sau là tên hệ thống tệp:

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

5. Làm cách nào để có thể sử dụng `mount` để gắn kết lại hệ thống tệp exFAT với UUID `6e2c12e3-472d-4bac-a257-c49ac07f3761` được gắn tại `/mnt/data` dưới dạng chỉ đọc?

   Vì hệ thống tệp đã được gắn kết nên bạn không cần phải lo lắng về loại hệ thống tệp hoặc ID, chỉ cần sử dụng tùy chọn `remount` với tham số `ro` (chỉ đọc) và điểm gắn kết:

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

6. Làm cách nào để có được danh sách tất cả các hệ thống tệp `ext3` và `ntfs` hiện được gắn trên một hệ thống?

   Sử dụng `mount -t`, theo sau là danh sách các hệ thống tệp được phân tách bằng dấu phẩy:

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

Đáp án Bài tập Mở rộng

1. Hãy xem xét mục sau trong `/etc/fstab`: `/dev/sdc1 /backup ext4 noatime,nouser,async`. Người dùng có thể gắn kết hệ thống tệp này bằng lệnh `mount /backup` không? Tại sao?

   Không, tham số `nouser` sẽ không cho phép người dùng thông thường gắn kết hệ thống tệp này.

2. Giả sử có một hệ thống tệp từ xa được gắn kết tại `/mnt/server`, hệ thống này hiện không thể truy cập được do mất kết nối mạng. Làm cách nào để có thể buộc nó phải ngắt gắn kết, hoặc nếu không thể thì được gắn ở dạng chỉ đọc?

   Truyền tham số `-f` và `-r` để ngắt gắn kết. Lệnh sẽ là `umount -f -r /mnt/server`. Hãy nhớ rằng bạn có thể nhóm các tham số; vì vậy, `umount -fr /mnt/server` cũng sẽ có tác dụng.

3. Hãy viết một mục nhập `/etc/fstab` sẽ gắn kết một ổ phân vùng btrfs với nhãn `Backup` trên `/mnt/backup` với các tùy chọn mặc định và không cho phép thực thi các tệp nhị phân từ nó.

   `LABEL=Backup /mnt/backup btrfs defaults,noexec`.

4. Hãy xem đơn vị gắn kết systemd sau:

   ```
   [Unit]
   Description=External data disk

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

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

   - Mục nhập `/etc/fstab` tương đương cho hệ thống tệp này là gì?

     Mục nhập sẽ là: `UUID=56C11DCC5D2E1334 /mnt/external ntfs defaults`

5. Tên tệp của đơn vị ở trên phải là gì để systemd có thể sử dụng nó? Nó nên được đặt ở đâu?

   Tên tệp phải giống với điểm gắn kết. Vì vậy, tên tệp sẽ là `mnt-external.mount` được đặt trong `/etc/systemd/system`.

104.5 Quản lý Quyền và Quyền sở hữu Tệp

Tham khảo các mục tiêu LPI

LPIC-1 version 5.0, Exam 101, Objective 104.5

Khối lượng

3

Các lĩnh vực kiến thức chính

- Quản lý quyền truy cập vào các tệp thông thường và đặc biệt cũng như các thư mục.
- Sử dụng các chế độ truy cập như suid, sgid và Sticky Bit để duy trì tính bảo mật.
- Biết cách thay đổi mặt nạ tạo tệp.
- Sử dụng trường nhóm để cấp quyền truy cập tệp cho các thành viên nhóm.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

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

Linux Professional Institute

104.5 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	104 Thiết bị, Hệ thống tệp Linux, Tiêu chuẩn phân cấp hệ thống Tệp
Mục tiêu:	104.5 Quản lý Quyền và Quyền sở hữu Tệp
Bài:	1 trên 1

Giới thiệu

Là một hệ thống đa người dùng, Linux cần một số cách để theo dõi người dùng nào sở hữu từng tệp nào và liệu họ có được phép thực hiện các thao tác trên tệp hay không. Điều này nhằm đảm bảo quyền riêng tư của những người dùng có thể muốn giữ bí mật nội dung tệp của họ cũng như để đảm bảo sự cộng tác bằng cách cho phép nhiều người dùng truy cập một số tệp nhất định.

Điều này được thực hiện thông qua hệ thống quyền ba cấp. Mỗi tệp trên đĩa đều thuộc quyền sở hữu của một người dùng và một nhóm người dùng và có ba bộ quyền: một dành cho chủ sở hữu của nó, một dành cho nhóm sở hữu tệp và một dành cho những người khác. Trong bài học này, chúng ta sẽ tìm hiểu cách truy vấn các quyền đối với một tệp, ý nghĩa của các quyền này và cách thao tác với chúng.

Truy vấn Thông tin về Tệp và Thư mục

Lệnh `ls` được sử dụng để lấy danh sách nội dung của bất kỳ một thư mục nào. Ở dạng cơ bản này, tất cả những gì ta sẽ nhận được là tên tệp:

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

Tuy nhiên, mỗi tệp đều sẽ có nhiều thông tin hơn thế, bao gồm loại, kích thước, quyền sở hữu và nhiều thông tin khác. Để xem thông tin này, ta phải yêu cầu `ls` cung cấp danh sách "chi tiết" bằng cách sử dụng tham số `-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
```

Mỗi cột ở đầu ra ở trên đều mang một ý nghĩa. Chúng ta hãy cùng xem xét các cột liên quan tới bài học này.

- Cột *đầu tiên* trên danh sách hiển thị loại tệp và quyền. Ví dụ: trên `drwxrwxr-x`:
  - Ký tự đầu tiên, `d`, cho biết loại tệp.
  - Ba ký tự tiếp theo, `rwx`, biểu thị quyền của chủ sở hữu tệp hay còn được gọi là *người dùng* (user) hoặc `u`.
  - Ba ký tự tiếp theo, `rwx`, cho biết quyền của *nhóm* (group) hoặc `g` sở hữu tệp .
  - Ba ký tự cuối cùng, `r-x`, biểu thị quyền dành cho *những người khác* (others) hoặc `o`.

TIP Chúng ta cũng sẽ thường thấy các nhóm quyền *khác* được gọi là quyền *thế giới*, tức là "Tất cả mọi người trên thế giới đều có những quyền này".

- Cột *thứ ba* và *thứ tư* hiển thị thông tin quyền sở hữu: tương ứng là người dùng và nhóm sở hữu tệp.
- Cột *thứ bảy* và cột cuối cùng hiển thị tên tệp.

Cột *thứ hai* cho biết số lượng liên kết cứng trỏ đến tệp đó. Cột *thứ năm* hiển thị kích thước tệp. Cột *thứ sáu* hiển thị ngày và giờ tệp được sửa đổi lần cuối. Nhưng những cột này không liên quan đến chủ đề hiện tại.

Còn Thư mục thì sao?

Nếu cố gắng truy vấn thông tin về một thư mục bằng cách sử dụng `ls -l`, nó sẽ hiển thị cho ta danh sách nội dung của thư mục:

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

Để tránh điều này và để truy vấn thông tin về chính thư mục đó, hãy thêm tham số `-d` vào `ls`:

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

Xem Tệp ẩn

Danh sách thư mục mà chúng ta đã truy xuất bằng `ls -l` trước đây chưa đầy đủ:

```
$ 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
```

Có ba tệp khác trong thư mục đó nhưng chúng bị ẩn. Trên Linux, các tệp có tên bắt đầu bằng dấu chấm (`.`) sẽ tự động bị ẩn. Để xem chúng, ta cần thêm tham số `-a` vào `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
```

Tệp `.thisIsHidden` bị ẩn đơn giản là vì tên của nó bắt đầu bằng `.`.

Tuy nhiên, các thư mục `.` và `..` là các thư mục đặc biệt. `.` là một con trỏ tới thư mục hiện tại và `..` là một con trỏ tới thư mục mẹ - thư mục chứa thư mục hiện tại. Trong Linux, mọi thư mục đều chứa ít nhất hai thư mục này.

TIP Ta có thể kết hợp nhiều tham số cho `ls` (và nhiều lệnh Linux khác). Ví dụ: `ls -l -a` có thể được viết thành `ls -la`.

Tìm hiểu về loại Tệp

Chúng ta đã đề cập tới việc chữ cái đầu tiên trong mỗi đầu ra của `ls -l` được dùng để mô tả loại tệp. Ba loại tệp phổ biến nhất là:

`-` (tệp bình thường)

Một tệp có thể chứa bất kỳ loại dữ liệu nào và giúp quản lý dữ liệu này. Các tệp có thể được sửa đổi, di chuyển, sao chép và xóa.

`d` (thư mục)

Một thư mục chứa các tệp hoặc thư mục khác và giúp tổ chức hệ thống tệp. Về mặt kỹ thuật, thư mục là một loại tệp đặc biệt.

`l` (liên kết tượng trưng)

"Tệp" này là một con trỏ tới một tệp hoặc thư mục khác ở nơi khác trong hệ thống tệp.

Ngoài những loại này còn có ít nhất ba loại tệp khác mà chúng ta nên biết nhưng nằm ngoài phạm vi của bài học này:

`b` (thiết bị khối)

Tệp này là viết tắt của một thiết bị ảo hoặc vật lý, thường là đĩa hoặc các loại thiết bị lưu trữ khác (chẳng hạn như đĩa cứng đầu tiên có thể được biểu thị bằng `/dev/sda`).

`c` (thiết bị ký tự)

Tệp này là viết tắt của một thiết bị ảo hoặc vật lý. Cửa sổ dòng lệnh (như cửa sổ dòng lệnh chính trên `/dev/ttyS0`) và các cổng nối tiếp là những ví dụ phổ biến về thiết bị ký tự.

`s` (ổ nối)

Ổ nối đóng vai trò là "ống dẫn" truyền thông tin giữa hai chương trình.

WARNING	Đừng thay đổi bất kỳ quyền nào trên thiết bị khối, thiết bị ký tự hoặc ổ nối trừ khi bạn biết chắc chắn mình đang làm gì. Điều này có thể khiến hệ thống dừng hoạt động!

Hiểu về Quyền

Trong đầu ra của `ls -l`, các quyền đối với tệp được hiển thị ngay sau loại tệp dưới dạng ba nhóm, mỗi nhóm có ba ký tự, theo thứ tự `r`, `w` và `x`. Dưới đây là ý nghĩa của chúng. Hãy nhớ rằng dấu gạch ngang `-` thể hiện việc thiếu quyền.

Quyền trên Tệp

`r`

Viết tắt của *read* (đọc) và có giá trị bát phân là `4` (đừng lo, chúng ta sẽ sớm thảo luận về khái niệm bát phân). Điều này đại diện cho quyền mở một tệp và đọc nội dung của nó.

`w`

Viết tắt của *write* (ghi) và có giá trị bát phân là `2`. Điều này đại diện cho quyền chỉnh sửa hoặc xóa một tệp.

`x`

Viết tắt của *execute* (thực thi) và có giá trị bát phân là `1`. Điều này đại diện cho việc tệp có thể được chạy dưới dạng tệp thực thi hoặc tệp lệnh.

Vì vậy, ví dụ: một tệp có quyền `rw-` có thể được đọc và ghi vào nhưng không thể được thực thi.

Quyền trên Thư mục

`r`

Viết tắt của *read* (đọc) và có giá trị bát phân là `4`. Nó đại diện cho quyền đọc nội dung của thư mục (ví dụ như tên tệp) nhưng nó *không* đại diện cho quyền đọc các tệp.

`w`

Viết tắt của *write* (ghi) và có giá trị bát phân là `2`. Nó đại diện cho quyền tạo hoặc xóa các tệp trong một thư mục.

Hãy nhớ rằng chúng ta không thể thực hiện những thay đổi này chỉ với quyền *ghi* mà còn cần có quyền *thực thi* (`x`) để thay đổi thư mục.

`x`

Viết tắt của *execute* (thực thi) và có giá trị bát phân là `1`. Nó đại diện cho quyền truy cập vào một thư mục nhưng không liệt kê các tệp của nó (để làm được việc này ta cần có `r`).

Phần cuối cùng về thư mục nghe có vẻ hơi khó hiểu. Ví dụ, hãy tưởng tượng rằng chúng ta có một thư mục có tên là `Another_Directory` với các quyền sau:

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

Cũng hãy tưởng tượng rằng bên trong thư mục này, chúng ta có một tệp lệnh vỏ tên là `hello.sh`:

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

Nếu chúng ta là người dùng `carol` và đang cố gắng liệt kê nội dung của `Another_Directory`, ta sẽ nhận được thông báo lỗi vì người dùng của chúng ta thiếu quyền đọc đối với thư mục đó:

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

Tuy nhiên, người dùng `carol` *có* quyền thực thi, có nghĩa là ta có thể truy cập vào thư mục. Do đó, người dùng `carol` có thể truy cập các tệp trong thư mục, miễn là người dùng này có quyền chính xác *đối với tệp tương ứng*. Giả sử người dùng có toàn quyền (`rwx`) đối với tệp lệnh `hello.sh`, họ sẽ *có thể* chạy tệp lệnh dù *không thể* đọc nội dung của thư mục chứa nó nếu họ biết tên tệp đầy đủ:

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

Như đã nói trước đây, các quyền được chỉ định theo trình tự: đầu tiên là cho chủ sở hữu tệp, sau đó là cho nhóm sở hữu và cuối cùng là những người dùng khác. Bất cứ khi nào có ai đó cố gắng thực hiện một hành động trên tệp, các quyền sẽ được kiểm tra theo thứ tự trên.

Trước tiên, hệ thống sẽ kiểm tra xem người dùng hiện tại có sở hữu tệp hay không, và nếu điều này đúng thì hệ thống chỉ áp dụng nhóm quyền đầu tiên. Nếu không, nó sẽ kiểm tra xem người dùng hiện tại có thuộc nhóm sở hữu tệp hay không. Trong trường hợp đó, nó sẽ chỉ áp dụng nhóm quyền thứ hai. Trong mọi trường hợp khác, hệ thống sẽ áp dụng nhóm quyền thứ ba.

Điều này có nghĩa là nếu người dùng hiện tại là chủ sở hữu của tệp thì chỉ có quyền của chủ sở hữu mới có hiệu lực ngay cả khi quyền nhóm hoặc quyền của những người khác mạnh hơn quyền của chủ sở hữu.

Sửa đổi Quyền của Tệp

Lệnh `chmod` được sử dụng để sửa đổi các quyền cho một tệp và có ít nhất hai tham số: tham số đầu tiên mô tả những quyền cần thay đổi và tham số thứ hai trỏ đến tệp hoặc thư mục nơi thay đổi sẽ được thực hiện. Hãy nhớ rằng chỉ chủ sở hữu tệp hoặc quản trị viên hệ thống (root) mới có thể thay đổi quyền trên tệp.

Quyền sửa đổi có thể được mô tả theo hai cách, hoặc "chế độ", khác nhau.

Chế độ đầu tiên được gọi là *chế độ tượng trưng*. Nó cung cấp khả năng kiểm soát chi tiết hơn, cho

phép ta thêm hoặc thu hồi một quyền duy nhất mà không sửa đổi các quyền khác. Chế độ còn lại được gọi là *chế độ bát phân*. Nó dễ nhớ hơn và nhanh hơn nếu ta muốn đặt tất cả các giá trị quyền cùng một lúc.

Cả hai chế độ sẽ dẫn đến cùng một kết quả cuối cùng. Vì vậy, các lệnh:

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

và

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

đều sẽ tạo ra cùng một đầu ra là một tệp có nhóm quyền như sau:

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

Bây giờ, hãy xem mỗi chế độ hoạt động như thế nào.

Chế độ Tượng trưng

Khi mô tả những quyền nào cần thay đổi trong *chế độ tượng trưng*, các ký tự đầu tiên sẽ cho biết các quyền mà ta cần thay đổi: quyền cho người dùng (`u`), cho nhóm (`g`), cho những người khác (`o`) và/hoặc cho tất cả mọi người (`a`).

Sau đó, ta cần cho lệnh biết phải làm gì: có thể cấp quyền (`+`), thu hồi quyền (`-`) hoặc đặt nó thành một giá trị cụ thể (`=`).

Cuối cùng, ta sẽ chỉ định quyền nào mình muốn đặt: đọc (`r`), ghi (`w`) hoặc thực thi (`x`).

Ví dụ: hãy tưởng tượng chúng ta có một tệp có tên `text.txt` với bộ quyền sau:

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

Nếu muốn cấp quyền ghi cho các thành viên trong nhóm sở hữu tệp, chúng ta sẽ sử dụng tham số `g+w`. Sẽ dễ dàng hơn nếu ta nghĩ về nó theo cách này: "Đối với nhóm (`g`), cấp (`+`) quyền ghi (`w`)". Vì vậy, lệnh sẽ là:

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

Hãy cùng kiểm tra kết quả với `ls`:

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

Nếu muốn xóa quyền đọc của chủ sở hữu của chính tệp đó, hãy nghĩ về nó như sau: "Đối với người dùng (`u`), thu hồi (`-`) quyền đọc (`r`)". Vậy tham số sẽ là `u-r` như sau:

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

Nếu muốn đặt quyền chính xác là `rw-` cho tất cả mọi người dùng? Hãy nghĩ về nó như sau: “Đối với tất cả mọi người (`a`), đặt chính xác (`=`) quyền đọc (`r`), quyền ghi (`w`), và không thực thi (`-`)”. Vì thế:

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

Tất nhiên, ta cũng có thể sửa đổi nhiều quyền cùng một lúc. Trong trường hợp này, hãy phân tách chúng bằng dấu phẩy (`,`):

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

Ví dụ trên có thể được đọc là: “Đối với người dùng (`u`), cấp (`+`) quyền đọc, ghi và thực thi (`rwx`); đối với nhóm (`g`), thu hồi (`-`) quyền thực thi (`x`)”.

Khi chạy trên một thư mục, `chmod` sẽ chỉ sửa đổi các quyền của thư mục. `chmod` cũng có chế độ đệ quy rất hữu ích khi ta muốn thay đổi quyền cho "tất cả các tệp trong một thư mục và các thư mục con của nó". Để sử dụng lệnh này, hãy thêm tham số `-R` sau tên lệnh và trước quyền cần thay đổi:

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

Lệnh này có thể được đọc là: "Đệ quy (`-R`), dành cho người dùng (`u`), cấp quyền (`+`) đọc, ghi và thực thi (`rwx`)".

WARNING Hãy cẩn thận và suy nghĩ kỹ trước khi sử dụng khóa chuyển `-R` vì nó có thể dễ dàng thay đổi quyền trên các tệp và thư mục mà bạn không muốn thay đổi, đặc biệt là trên các thư mục có số lượng tệp và thư mục con lớn.

Chế độ Bát phân

Trong *chế độ bát phân*, các quyền sẽ được chỉ định theo một cách khác: dưới dạng giá trị ba chữ số trên ký hiệu bát phân - hệ thống cơ số 8.

Mỗi quyền đều có một giá trị tương ứng và chúng được chỉ định theo thứ tự sau: đầu tiên là đọc (`r`) được biểu thị bằng `4`, sau đó là ghi (`w`) được biểu thị bằng `2` và cuối cùng là thực thi (`x`) được biểu thị bằng `1`. Nếu không có quyền, hãy sử dụng giá trị 0 (`0`). Vì vậy, quyền của `rwx` sẽ là `7` (`4+2+1`) và `r-x` sẽ là `5` (`4+0+1`).

Chữ số đầu tiên trong ba chữ số trên bộ quyền thể hiện các quyền dành cho người dùng (`u`), chữ số thứ hai dành cho nhóm (`g`) và chữ số thứ ba dành cho những người khác (`o`). Nếu muốn đặt quyền cho một tệp thành `rw-rw----`, giá trị bát phân sẽ là `660`:

```
$ chmod 660 text.txt
$ ls -l text.txt
-rw-rw---- 1 carol carol 765 Dec 20 21:25 text.txt
```

Ngoài ra, cú pháp trong *chế độ bát phân* cũng giống như trong *chế độ tượng trưng*, tham số đầu tiên thể hiện các quyền ta muốn thay đổi và tham số thứ hai trỏ đến tệp hoặc thư mục nơi thay đổi sẽ được thực hiện.

TIP Nếu một giá trị quyền là *số lẻ* thì tệp chắc chắn có thể thực thi được!

Ta nên sử dụng cú pháp nào? Nếu muốn thay đổi quyền thành một giá trị cụ thể như `640` (`rw-r-- ---`), hãy sử dụng *chế độ bát phân*.

Nếu chỉ muốn thay đổi một giá trị cụ thể bất kể các quyền hiện tại đối với tệp là gì, chế độ *tượng trưng* sẽ hữu ích hơn. Ví dụ: ta có thể thêm quyền thực thi cho người dùng chỉ bằng cách sử dụng `chmod u+x script.sh` mà không cần quan tâm đến hoặc thậm chí chạm vào các quyền hiện tại cho nhóm và những người khác.

Sửa đổi Quyền sở hữu Tệp

Lệnh `chown` được sử dụng để sửa đổi quyền sở hữu của một tệp hoặc thư mục. Cú pháp của lệnh khá đơn giản:

```
chown USERNAME:GROUPNAME FILENAME
```

Ví dụ: chúng ta cùng xem một tệp có tên `text.txt`:

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

Người dùng sở hữu tệp là `carol` và nhóm cũng là `carol`. Bây giờ, chúng ta sẽ thay đổi nhóm sở hữu tệp thành một nhóm khác là `students`:

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

Hãy nhớ rằng người dùng sở hữu tệp không cần phải thuộc nhóm sở hữu tệp. Trong ví dụ trên, người dùng `carol` không cần phải là thành viên của nhóm `students`.

Bộ quyền của người dùng hoặc nhóm có thể được bỏ qua nếu ta không muốn thay đổi chúng. Vì vậy, để chỉ thay đổi nhóm sở hữu một tệp, ta sẽ sử dụng `chown :students text.txt`. Để chỉ thay đổi người dùng, lệnh sẽ là `chown carol: text.txt` hoặc `chown carol text.txt`. Ngoài ra, ta cũng có thể sử dụng lệnh `chgrp students text.txt`.

Trừ khi là quản trị viên hệ thống (root), chúng ta sẽ không thể thay đổi quyền sở hữu tệp cho người dùng hoặc nhóm khác mà mình không thuộc về. Nếu cố gắng thực hiện việc này, ta sẽ nhận được thông báo lỗi `Operation not permitted` (Hoạt động không được phép).

Truy vấn Nhóm

Trước khi thay đổi quyền sở hữu một tệp, việc biết nhóm nào tồn tại trên hệ thống, người dùng nào là thành viên của nhóm và người dùng thuộc nhóm nào có thể sẽ rất hữu ích.

Để xem nhóm nào tồn tại trên hệ thống, hãy nhập `getent group`. Đầu ra sẽ tương tự như sa (đầu ra đã được tóm gọn):

```
$ 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
```

Nếu muốn biết một người dùng thuộc nhóm nào, hãy thêm tên người dùng làm tham số cho `groups`:

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

Để làm ngược lại (xem những người dùng nào thuộc về một nhóm), hãy sử dụng `groupmems`. Tham số `-g` chỉ định nhóm và `-l` sẽ liệt kê tất cả các thành viên của nhóm:

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

TIP `groupmems` chỉ có thể được chạy với quyền gốc - tức quyền của quản trị viên hệ thống. Nếu hiện chưa đăng nhập vào root, hãy thêm `sudo` trước lệnh.

Quyền mặc định

Hãy cùng thử một thí nghiệm: mở một cửa sổ dòng lệnh và tạo một tệp trống bằng lệnh sau:

```
$ touch testfile
```

Bây giờ, hãy xem các quyền cho tệp này. Tuỳ từng hệ thống mà kết quả *có thể* sẽ khác, nhưng chúng ta hãy giả sử chúng trông giống như sau:

```
$ ls -lh testfile
```

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

Các quyền là `rw-r—r--`: *đọc* và *ghi* cho người dùng và *đọc* cho nhóm và những người khác (hoặc `644` ở chế độ bát phân). Bây giờ, hãy thử tạo một thư mục:

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

Bây giờ, các quyền là `rwxr-xr-x` : *đọc*, *ghi* và *thực thi* cho người dùng, *đọc* và *thực thi* cho nhóm và những người khác (`755` ở chế độ bát phân).

Bất kể là đang ở đâu trong hệ thống tệp, mọi tệp hoặc thư mục chúng ta tạo ra đều sẽ nhận được các quyền tương tự. Bạn có bao giờ tự hỏi xem chúng đến từ đâu không?

Chúng đến từ *usermask* hoặc `umask` dùng để đặt quyền mặc định cho mọi tệp được tạo. Ta có thể kiểm tra các giá trị hiện tại bằng lệnh `umask`:

```
$ umask
0022
```

Nhưng nó trông không giống như `rw-r—r--` hay thậm chí là `644`. Có lẽ chúng ta nên thử với tham số `-S` để có được đầu ra ở chế độ tượng trưng:

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

Đó là những quyền tương tự mà thư mục ví dụ của chúng ta. Nhưng tại sao khi chúng ta tạo một tệp thì các quyền lại khác nhau?

Theo mặc định, thật vô nghĩa khi đặt quyền thực thi chung cho tất cả mọi người trên bất kỳ một tệp nào. Các thư mục cần có quyền thực thi (nếu không thì người dùng không thể truy cập chúng) nhưng các tệp thì không nên, vì thế mà chúng sẽ không có các quyền thực thi. Do đó mà ta có `rw-r—r--`.

Bên cạnh việc hiển thị các quyền mặc định, `umask` cũng có thể được sử dụng để thay đổi chúng cho phiên vỏ hiện tại. Ví dụ: nếu chúng ta sử dụng lệnh:

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

Mọi thư mục mới sẽ kế thừa các quyền `rwxrwx---` và mọi tệp `rw-rw----` (vì chúng không có quyền thực thi). Nếu lặp lại các ví dụ trên để tạo một `testfile` và `testdir` và kiểm tra các quyền, ta sẽ nhận được:

```
$ 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
```

Và nếu kiểm tra `umask` mà không có tham số `-S` (chế độ tượng trưng) đi kèm, ta sẽ nhận được:

```
$ umask
0007
```

Kết quả trông hơi lạ vì các giá trị được sử dụng là khác nhau. Dưới đây là bảng gồm tất cả các giá trị và ý nghĩa tương ứng của chúng:

Giá trị	Quyền cho Tệp	Quyền cho Thư mục
`0`	`rw-`	`rwx`
`1`	`rw-`	`rw-`
`2`	`r--`	`r-x`
`3`	`r--`	`r--`
`4`	`-w-`	`-wx`
`5`	`-w-`	`-w-`
`6`	`---`	`--x`
`7`	`---`	`---`

Như có thể thấy được, `007` tương ứng với `rwxrwx---`, chính xác như những gì chúng ta đã yêu cầu. Số 0 đứng đầu có thể được bỏ qua.

Quyền đặc biệt

Ngoài các quyền đọc, ghi và thực thi cho người dùng, nhóm và những người khác, mỗi tệp có thể có ba *quyền đặc biệt* khác có thể thay đổi cách hoạt động của một thư mục hoặc cách một chương trình chạy. Chúng có thể được chỉ định ở chế độ tượng trưng hoặc bát phân như sau:

Bit dính

Bit dính (còn được gọi là *cờ xóa hạn chế*) có giá trị bát phân là `1` và ở chế độ tượng trưng được biểu thị bằng `t` trong phạm vi quyền của những người khác. Điều này chỉ áp dụng cho các thư mục và không ảnh hưởng đến các tệp thông thường. Trên Linux, nó sẽ ngăn người dùng xóa hoặc đổi tên tệp trong một thư mục trừ khi họ sở hữu tệp hoặc thư mục đó.

Các thư mục có bộ bit dính sẽ hiển thị `t` thay thế `x` ở quyền cho *những người khác* trên đầu ra của `ls -l`:

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

Ở chế độ bát phân, các quyền đặc biệt được chỉ định bằng ký hiệu 4 chữ số với chữ số đầu tiên thể hiện quyền đặc biệt cần thay đổi. Ví dụ: để đặt bit dính (giá trị `1`) cho thư mục `Another_Directory` ở chế độ bát phân với quyền `755`, lệnh sẽ là:

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

Set GID

Set GID (còn được gọi là bit SGID hoặc Thiết lập ID Nhóm) có giá trị bát phân là `2` và ở chế độ tượng trưng được biểu thị bằng `s` trên quyền *nhóm*. Nó có thể được áp dụng cho các tệp hoặc thư mục thực thi. Trên các tệp, nó sẽ làm cho tiến trình chạy với các đặc quyền của nhóm sở hữu tệp. Khi áp dụng cho các thư mục, nó sẽ làm cho mọi tệp hoặc thư mục được tạo trong đó kế thừa nhóm từ thư mục mẹ.

Các tệp và thư mục có bit SGID sẽ hiển thị `s` thay thế `x` trên các quyền cho *nhóm* trên đầu ra của `ls -l`:

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

Để thêm quyền SGID vào tệp ở chế độ tượng trưng, lệnh sẽ là:

```
$ chmod g+s test.sh
$ ls -l test.sh
```

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

Ví dụ sau sẽ giúp ta hiểu rõ hơn về tác dụng của SGID trên một thư mục. Giả sử chúng ta có một thư mục tên là `Sample_Directory` được sở hữu bởi người dùng `carol` và nhóm `users` với cấu trúc quyền sau:

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

Bây giờ, hãy thay đổi thư mục này và sử dụng lệnh `touch` để tạo một tệp trống bên trong nó. Kết quả sẽ là:

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

Như chúng ta có thể thấy, tệp này thuộc sở hữu của người dùng `carol` và nhóm `carol`. Tuy nhiên, nếu thư mục có quyền SGID thì kết quả sẽ khác. Đầu tiên, chúng ta thêm bit SGID vào `Sample_Directory` và kiểm tra kết quả:

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

Ký tự `s` trên quyền của nhóm cho biết bit SGID đã được thiết lập. Bây giờ, chúng ta sẽ thay đổi thư mục này và một lần nữa tạo một tệp trống bằng lệnh `touch`:

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

Nhóm sở hữu tệp là `users`. Điều này là do bit SGID đã làm cho tệp kế thừa chủ sở hữu nhóm của thư mục mẹ của nó là `users`.

Set UID

SUID (còn được gọi là Thiết lập ID Người dùng) có giá trị bát phân là `4` và được biểu thị bằng `s` trên quyền *người dùng* ở chế độ tượng trưng. Nó chỉ áp dụng cho các tệp và không có tác dụng đối với các thư mục. Hành vi của nó tương tự như bit SGID nhưng tiến trình sẽ chạy với đặc quyền của *người dùng* sở hữu tệp. Các tệp có bit SUID sẽ hiển thị `s` thay thế `x` trên các quyền cho người dùng ở đầu ra của `ls -l`:

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

Ta có thể kết hợp nhiều quyền đặc biệt trên một tham số. Vì vậy, để đặt SGID (giá trị `2`) và SUID (giá trị `4`) ở chế độ bát phân cho tệp lệnh `test.sh` có quyền `755`, ta sẽ gõ:

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

Và kết quả sẽ là:

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

TIP

Nếu cửa sổ dòng lệnh của bạn hỗ trợ màu (ngày nay hầu hết các hệ thống đều có hỗ trợ), bạn có thể nhanh chóng xem được liệu các quyền đặc biệt này có được thiết lập hay không bằng cách xem lướt qua đầu ra của `ls -l`. Đối với bit dính, tên thư mục có thể được hiển thị bằng phông chữ màu đen với nền xanh lam. Điều tương tự cũng áp dụng cho các tệp có bit SGID (nền vàng) và SUID (nền đỏ). Màu sắc có thể sẽ khác tùy thuộc vào cài đặt cửa sổ dòng lệnh và bản phân phối Linux mà bạn sử dụng.

Bài tập Hướng dẫn

1. Hãy tạo một thư mục có tên `emptydir` bằng lệnh `mkdir Emptydir`. Bây giờ, bằng cách sử dụng `ls`, hãy liệt kê các quyền cho thư mục `emptydir`.

2. Hãy tạo một tệp trống có tên `emptyfile` bằng lệnh `touch Emptyfile`. Bây giờ, bằng cách sử dụng `chmod` ở chế độ tượng trưng, hãy thêm quyền thực thi cho chủ sở hữu tệp `emptyfile`, đồng thời xóa quyền ghi và thực thi đối với những người khác. Hãy thực hiện việc này chỉ bằng lệnh `chmod`.

3. Các quyền mặc định cho một tệp sẽ là gì nếu giá trị `umask` được đặt thành `027`?

4. Giả sử một tệp có tên `test.sh` là tệp lệnh vỏ có các quyền và quyền sở hữu sau:

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

   - Quyền của chủ sở hữu tệp là gì?

   - Sử dụng ký hiệu bát phân, cú pháp của `chmod` sẽ là gì để "bỏ thiết lập" quyền đặc biệt được cấp cho tệp này?

5. Hãy xem tệp sau:

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

   `sdb1` là loại tệp nào? Ai có thể ghi vào nó?

6. Hãy xem 4 tệp sau:

```
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
```

Hãy viết ra các quyền tương ứng cho từng tệp và thư mục bằng chế độ bát phân sử dụng ký hiệu 4 chữ số.

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

Bài tập Mở rộng

1. Hãy thử điều này trên cửa sổ dòng lệnh: tạo một tệp trống có tên `emptyfile` bằng lệnh `touchemptyfile`. Bây giờ, hãy "loại hết" các quyền đối với tệp có `chmod 000 Emptyfile`. Điều gì sẽ xảy ra nếu bạn thay đổi quyền cho `emptyfile` bằng cách chỉ truyền *một* giá trị cho `chmod` ở chế độ bát phân (như `chmod 4 Emptyfile`)? Và nếu bạn sử dụng hai chữ số (như trong `chmod 44emptyfile`) thì sao? Chúng ta có thể hiểu được điều gì về cách `chmod` đọc giá trị số?

2. Hãy xem xét các quyền cho thư mục tạm thời trên một hệ thống Linux, `/tmp`:

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

   Người dùng, nhóm và những người khác có toàn quyền nhưng người dùng thông thường có thể xóa *bất kỳ* một tệp nào trong thư mục này không? Tại sao lại như vậy?

3. Một tệp có tên `test.sh` có các quyền `-rwsr-xr-x`, nghĩa là bit SUID được thiết lập. Bây giờ, hãy chạy các lệnh sau:

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

   Chúng ta đã làm gì? Chữ hoa `S` có nghĩa là gì?

4. Làm cách nào để có thể tạo một thư mục có tên `Box` mà trong đó tất cả các tệp đều được tự động sở hữu bởi nhóm `users` và chỉ người dùng đã tạo ra chúng mới có thể xóa chúng?

Tóm tắt

Trong bài học này, chúng ta đã học cách sử dụng `ls` để lấy (và giải mã) thông tin về quyền truy cập tệp, cách kiểm soát hoặc thay đổi người có thể tạo, xóa hoặc sửa đổi tệp bằng `chmod` cả ở chế độ *bát phân* và *tượng trưng*, cách để thay đổi quyền sở hữu các tệp bằng `chown` và `chgrp` cũng như cách truy vấn và thay đổi mặt nạ quyền mặc định cho các tệp và thư mục bằng `umask`.

Các lệnh sau đã được thảo luận trong bài học này:

`ls`

Liệt kê các tệp, có thể tùy chọn bao gồm các chi tiết như quyền.

`chmod`

Thay đổi quyền của một tệp hoặc thư mục.

`chown`

Thay đổi người dùng và/hoặc nhóm sở hữu tệp hoặc thư mục.

`chgrp`

Thay đổi nhóm sở hữu của một tệp hoặc thư mục.

`umask`

Truy vấn hoặc thiết lập mặt nạ quyền mặc định cho tệp và thư mục

Đáp án Bài tập Hướng dẫn

1. Hãy tạo một thư mục có tên `emptydir` bằng lệnh `mkdir Emptydir`. Bây giờ, bằng cách sử dụng `ls`, hãy liệt kê các quyền cho thư mục `emptydir`.

   Thêm tham số `-d` vào `ls` để xem thuộc tính tệp của một thư mục thay vì liệt kê nội dung của nó. Vì vậy, câu trả lời sẽ là:

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

   Một điểm cộng nếu bạn hợp nhất hai tham số thành một (như trong `ls -ld Emptydir`).

2. Hãy tạo một tệp trống có tên `emptyfile` bằng lệnh `touch Emptyfile`. Bây giờ, bằng cách sử dụng `chmod` ở chế độ tượng trưng, hãy thêm quyền thực thi cho chủ sở hữu tệp `emptyfile`, đồng thời xóa quyền ghi và thực thi đối với những người khác. Hãy thực hiện việc này chỉ bằng lệnh `chmod`.

   Hãy nghĩ theo cách sau:

   - "Đối với người dùng sở hữu tệp (`u`) hãy thêm (`+`) quyền thực thi (`x`)", vì vậy nên ta có `u+x`.
   - "Đối với nhóm (`g`) và những người dùng khác (`o`), hãy xóa (`-`) quyền ghi (`w`) và thực thi (`x`)", vì vậy nên ta có `go-wx`.

     Để kết hợp hai bộ quyền này, ta sẽ thêm dấu phẩy vào giữa chúng. Vì vậy, kết quả cuối cùng là:

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

3. Các quyền mặc định cho một tệp sẽ là gì nếu giá trị `umask` được đặt thành `027`?

   Các quyền sẽ là `rw-r------`

4. Giả sử một tệp có tên `test.sh` là tệp lệnh vỏ có các quyền và quyền sở hữu sau:

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

   - Quyền của chủ sở hữu tệp là gì?

     Các quyền dành cho chủ sở hữu (ký tự thứ 2 đến thứ 4 trong đầu ra của `ls -l`) là `rwx`. Vì vậy, câu trả lời sẽ là: "đọc, ghi và thực thi tệp".

- Sử dụng ký hiệu bát phân, cú pháp của chmod để "huỷ thiết lập" quyền đặc biệt được cấp cho tệp này là gì?

  Chúng ta có thể "huỷ thiết lập" các quyền đặc biệt bằng cách truyền chữ số thứ 4, `0` cho `chmod`. Các quyền hiện tại là `755`. Vì vậy, lệnh phải là `chmod 0755`.

5. Hãy xem tệp sau:

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

   `sdb1` là loại tệp nào? Ai có thể ghi vào nó?

   Ký tự đầu tiên ở đầu ra của `ls -l` sẽ hiển thị loại tệp. `b` là một *thiết bị khối*, thường là một đĩa (cục bộ hoặc ngoại vi) được kết nối với máy. Chủ sở hữu (`root`) và bất kỳ người dùng nào trong nhóm `disk` đều có thể ghi vào đó.

6. Hãy xem 4 tệp sau:

```
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
```

   Hãy viết ra các quyền tương ứng cho từng tệp và thư mục bằng chế độ bát phân sử dụng ký hiệu 4 chữ số.

   Các quyền tương ứng ở chế độ bát phân như sau:

`Another_Directory`	`1755`. `1` cho bit dính, `755` cho các quyền thông thường (`rwx` cho người dùng, `r-x` cho nhóm và những người khác).
`foo.bar`	`0044`. Không có quyền đặc biệt (vì vậy nên chữ số đầu tiên là `0`), không có quyền cho người dùng (`---`) và chỉ có quyền đọc (`r–r--`) cho nhóm và những người khác.
`HugeFile.zip`	`0664`. Không có quyền đặc biệt nên chữ số đầu tiên là `0`. `6` (`rw-`) cho người dùng và nhóm, `4` (`r--`) cho những người khác.

`Sample_Directory`	`2755`. `2` cho bit SGID, `7` (`rwx`) cho người dùng, `5` (`r-x`) cho nhóm và những người khác.

Đáp án Bài tập Mở rộng

1. Hãy thử điều này trên cửa sổ dòng lệnh: tạo một tệp trống có tên `emptyfile` bằng lệnh `touchemptyfile`. Bây giờ, hãy "loại hết" các quyền đối với tệp có `chmod 000 Emptyfile`. Điều gì sẽ xảy ra nếu bạn thay đổi quyền cho `emptyfile` bằng cách chỉ truyền *một* giá trị cho `chmod` ở chế độ bát phân (như `chmod 4 Emptyfile`)? Và nếu bạn sử dụng hai chữ số (như trong `chmod 44emptyfile`) thì sao? Chúng ta có thể hiểu được điều gì về cách `chmod` đọc giá trị số?

   Hãy nhớ rằng chúng tađã "bỏ hết" các quyền đối với `emptyfile`. Vì vậy, trạng thái ban đầu của nó sẽ là:

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

   Bây giờ, chúng ta hãy thử lệnh đầu tiên là `chmod 4 Emptyfile`:

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

   Quyền của *những người khác* đã được thay đổi. Và điều gì sẽ xảy ra nếu chúng ta thử hai chữ số (như trong `chmod 44emptyfile`)?

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

   Bây giờ, quyền của *nhóm* và *những người khác* đã bị ảnh hưởng. Từ đó, chúng ta có thể kết luận rằng ở chế độ bát phân, `chmod` sẽ đọc giá trị "ngược", từ chữ số có ít ý nghĩa nhỏ nhất (*những người khác*) đến chữ số có nhiều ý nghĩa nhất (*người dùng*). Nếu truyền một chữ số, bạn sẽ sửa đổi quyền cho *những người khác*. Với hai chữ số, bạn sẽ sửa đổi quyền cho *nhóm* và *những người khác*. Với ba chữ số, bạn sẽ sửa đổi quyền cho *người dùng*, *nhóm* và *những người khác*. Với bốn chữ số, bạn sẽ sửa đổi quyền cho *người dùng*, *nhóm*, *những người khác* và các quyền đặc biệt.

2. Hãy xem xét các quyền cho thư mục tạm thời trên một hệ thống Linux, `/tmp`:

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

Người dùng, nhóm và những người khác có toàn quyền nhưng người dùng thông thường có thể xóa *bất kỳ* một tệp nào trong thư mục này không? Tại sao lại như vậy?

`/tmp` là cái mà chúng ta gọi là thư mục *cả thế giới có thể ghi được*, nghĩa là bất kỳ người dùng nào cũng có thể ghi vào đó. Nhưng chúng ta không muốn một người dùng nào đó làm xáo trộn các tệp do những người khác tạo ra, vì vậy nên *bit dính* sẽ được đặt (như được biểu thị bằng `t` trên các quyền dành cho *những người khác*). Điều này có nghĩa là người dùng có thể xóa các tệp trên `/tmp` (nhưng chỉ những tệp do chính họ tạo ra).

3. Một tệp có tên `test.sh` có các quyền `-rwsr-xr-x`, nghĩa là bit SUID được thiết lập. Bây giờ, hãy chạy các lệnh sau:

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

Chúng ta đã làm gì? Chữ hoa `S` có nghĩa là gì?

Chúng ta đã xóa quyền thực thi đối với người dùng sở hữu tệp. `s` (hoặc `t`) sẽ thay thế `x` trên đầu ra của `ls -l`, vì vậy nên hệ thống cần một cách để hiển thị xem người dùng có quyền thực thi hay không. Nó thực hiện điều này bằng cách thay đổi kiểu chữ của ký tự đặc biệt.

Chữ `s` trong nhóm quyền đầu tiên có nghĩa là người dùng sở hữu tệp có quyền thực thi và bit SUID đã được thiết lập. Chữ `S` có nghĩa là người dùng sở hữu tệp thiếu (`-`) quyền thực thi và bit SUID đã được thiết lập.

Điều tương tự cũng có thể xảy ra với SGID. Chữ `s` trên nhóm quyền thứ hai có nghĩa là nhóm sở hữu tệp có quyền thực thi và bit SGID đã được thiết lập. Chữ `S` có nghĩa là nhóm sở hữu tệp thiếu (`-`) quyền thực thi và bit SGID đã được thiết lập.

Điều này cũng đúng với bit dính được biểu thị bằng `t` trên nhóm quyền thứ ba. Chữ `t` có nghĩa là bộ bit dính và những người khác có quyền thực thi. Chữ `T` có nghĩa là bit dính đã được thiết lập và những người khác không có quyền thực thi.

4. Làm cách nào để có thể tạo một thư mục có tên `Box` mà trong đó tất cả các tệp đều được sở hữu tự động bởi nhóm `users` và chỉ người dùng đã tạo ra chúng mới có thể xóa chúng?

Đây là một quá trình gồm nhiều bước. Bước đầu tiên là tạo thư mục:

```
$ mkdir Box
```

Chúng ta muốn mọi tệp được tạo trong thư mục này sẽ được tự động gán cho nhóm `users`. Ta có thể thực hiện điều này bằng cách đặt nhóm này làm chủ sở hữu của thư mục và sau đó bằng cách thiết lập bit SGID trên đó. Chúng ta cũng cần đảm bảo rằng bất kỳ thành viên nào trong nhóm cũng đều có thể ghi vào thư mục đó.

Vì không quan tâm đến các quyền khác là gì và chỉ muốn "thay đổi" các bit đặc biệt nên chúng ta sẽ sử dụng chế độ tượng trưng:

```
$ chown :users Box/
$ chmod g+wxs Box/
```

Hãy lưu ý rằng nếu người dùng hiện tại của bạn không thuộc nhóm `users`, bạn sẽ phải sử dụng lệnh `sudo` trước các lệnh trên để thực hiện thay đổi với quyền gốc.

Bây giờ là bước cuối cùng, hãy đảm bảo rằng chỉ người dùng đã tạo tệp mới được phép xóa tệp đó. Điều này được thực hiện bằng cách thiết lập bit dính (được biểu thị bằng `t`) trên thư mục. Hãy nhớ rằng nó được đặt trên các quyền dành cho người khác (`o`).

```
$ chmod o+t Box/
```

Các quyền trên thư mục `Box` phải như sau:

```
drwxrwsr-t 2 carol users 4,0K Jan 18 19:09 Box
```

Tất nhiên, bạn có thể chỉ định SGID và bit dính chỉ bằng một lệnh `chmod`:

```
$ chmod g+wxs,o+t Box/
```

Một điểm cộng nếu bạn đã nghĩ đến điều này.

104.6 Tạo và thay đổi các Liên kết cứng và tượng trưng

Tham khảo các mục tiêu LPI

LPIC-1 version 5.0, Exam 101, Objective 104.6

Khối lượng

2

Các lĩnh vực kiến thức chính

- Tạo liên kết.
- Xác định các liên kết cứng và/hoặc mềm.
- Sao chép so với liên kết tệp.
- Sử dụng liên kết để hỗ trợ các công việc quản trị hệ thống.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

- `ln`
- `ls`

Linux Professional Institute

104.6 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	104 Thiết bị, Hệ thống tệp Linux, Tiêu chuẩn phân cấp hệ thống Tệp
Mục tiêu:	104.6 Tạo và Thay đổi các Liên kết Cứng và Tượng trưng
Bài:	1 trên 1

Giới thiệu

Trong Linux, một số tệp sẽ được xử lý đặc biệt do vị trí lưu trữ của chúng (chẳng hạn như các tệp tạm thời) hoặc cách chúng tương tác với hệ thống tệp (như các liên kết). Trong bài học này, chúng ta sẽ tìm hiểu liên kết là gì và cách quản lý chúng.

Hiểu về Liên kết

Như đã nói, trên Linux, mọi thứ đều được coi là một tệp. Nhưng có một loại tệp đặc biệt được gọi là *liên kết* và có hai loại liên kết trên hệ thống Linux:

Liên kết tượng trưng

Còn được gọi là *liên kết mềm*. Chúng trỏ đến đường dẫn của tệp khác. Nếu ta xóa tệp mà liên kết trỏ đến (được gọi là *đích*), liên kết vẫn tồn tại nhưng sẽ "ngừng hoạt động" vì nó đang không trỏ đến "bất cứ thứ gì".

Liên kết cứng

Hãy coi liên kết cứng là cái tên thứ hai của tệp gốc. Chúng *không* trùng lặp mà thay vào đó là một mục bổ sung trong hệ thống tệp trỏ đến cùng một vị trí (nút định danh) trên đĩa.

TIP

nút định danh (inode) là một cấu trúc dữ liệu lưu trữ các thuộc tính cho một đối tượng (như tệp hoặc thư mục) trên hệ thống tệp. Trong số các thuộc tính đó có quyền, quyền sở hữu và dữ liệu của đối tượng được lưu trữ trên khối đĩa nào. Hãy coi nó như một mục nhập trên một chỉ mục, từ đó mà nó có tên như vậy (bắt nguồn từ "index node").

Làm việc với các Liên kết Cứng

Tạo Liên kết Cứng

Lệnh tạo liên kết cứng trên Linux là `ln`. Cú pháp cơ bản là:

```
$ ln TARGET LINK_NAME
```

`TARGET` phải tồn tại sẵn (đây là tệp mà liên kết sẽ trỏ tới) và nếu mục tiêu không có trong thư mục hiện tại hoặc nếu muốn tạo liên kết ở nơi khác, ta sẽ *phải* chỉ định đường dẫn đầy đủ đến nó. Ví dụ: lệnh

```
$ ln target.txt /home/carol/Documents/hardlink
```

sẽ tạo một tệp có tên là `hardlink` trong thư mục `/home/carol/Documents/` được liên kết với tệp `target.txt` trong thư mục hiện tại.

Nếu bỏ qua tham số cuối cùng (`LINK_NAME`), một liên kết có cùng tên với mục tiêu sẽ được tạo trong thư mục hiện tại.

Quản lý Liên kết Cứng

Liên kết cứng là các mục trong hệ thống tệp có tên khác nhau nhưng trỏ đến cùng một dữ liệu trên đĩa. Tất cả các tên như vậy đều ngang nhau và có thể được sử dụng để chỉ một tệp. Nếu ta thay đổi nội dung của một trong các tên, nội dung của tất cả các tên khác trỏ đến tệp đó cũng sẽ thay đổi vì tất cả các tên này đều trỏ đến cùng một dữ liệu. Nếu ta xóa một tên thì các tên khác vẫn sẽ hoạt động.

Điều này xảy ra là do khi ta "xóa" một tệp, dữ liệu không thực sự bị xóa khỏi đĩa. Hệ thống sẽ chỉ xóa mục nhập trên bảng hệ thống tệp trỏ đến nút định danh tương ứng với dữ liệu trên đĩa. Nhưng nếu có một mục nhập thứ hai trỏ đến cùng một nút định danh, chúng ta vẫn có thể truy

cập được dữ liệu. Hãy coi nó như hai con đường hội tụ tại một điểm. Ngay cả khi ta chặn hoặc chuyển hướng một trong hai con đường thì vẫn có thể đến đích bằng con đường còn lại.

Chúng ta có thể kiểm tra điều này bằng cách sử dụng tham số `-i` của `ls`. Hãy xem xét các nội dung sau đây của một thư mục:

```
$ 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
```

Số đứng trước quyền chính là số nút định danh. Ta có thể thấy rằng cả tệp `hardlink` và tệp `target.txt` đều có cùng số định danh (`3806696`). Điều này là do tệp này chính là liên kết cứng của tệp kia.

Nhưng cái nào là tệp gốc và cái nào là liên kết? Chúng ta sẽ không thể biết được vì chúng luôn tương đương nhau trong tất cả các mục đích thực tế.

Hãy lưu ý rằng mọi liên kết cứng trỏ đến một tệp sẽ tăng *số lượng liên kết* của tệp. Đây là con số đứng ngay sau các quyền trong đầu ra của `ls -l`. Theo mặc định, mọi tệp đều có số lượng liên kết là `1` (các thư mục có số lượng là `2`) và mọi liên kết cứng trỏ tới nó sẽ tăng số lượng lên một. Đó là lý do số lượng liên kết là `2` trên các tệp trong danh sách ở trên.

Ngược lại với các liên kết tượng trưng, chúng ta chỉ có thể tạo các liên kết cứng tới các tệp, và cả liên kết và đích đều phải nằm trong cùng một hệ thống tệp.

Di chuyển và loại bỏ các Liên kết Cứng

Vì các liên kết cứng cũng được coi là các tệp thông thường nên chúng có thể bị xóa bằng `rm` và có thể bị đổi tên hoặc di chuyển trên khắp hệ thống tệp bằng `mv`. Và vì một liên kết cứng sẽ trỏ đến cùng một nút định danh với đích nên nó có thể được di chuyển tự do mà không sợ liên kết bị "phá vỡ".

Liên kết Tượng trưng

Tạo Liên kết Tượng trưng

Lệnh được sử dụng để tạo một liên kết tượng trưng cũng là `ln` nhưng sẽ có thêm tham số `-s` như sau:

```
$ ln -s target.txt /home/carol/Documents/softlink
```

Thao tác này sẽ tạo một tệp có tên là `softlink` trong thư mục `/home/carol/Documents/` trỏ tới tệp `target.txt` trong thư mục hiện tại.

Cũng giống như liên kết cứng, chúng ta có thể bỏ qua tên liên kết để tạo liên kết có cùng tên với đích trong thư mục hiện tại.

Quản lý Liên kết Tượng trưng

Các liên kết tượng trưng sẽ trỏ đến một đường dẫn khác trong hệ thống tệp. Chúng ta có thể tạo các liên kết mềm đến các tệp *và* thư mục ngay cả trên các phân vùng khác nhau. Khá dễ dàng để có thể phát hiện một liên kết tượng trưng trong đầu ra của `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
```

Trong ví dụ trên, ký tự đầu tiên trên các quyền đối với tệp `softlink` là `l` biểu thị một liên kết tượng trưng. Hơn nữa, ngay sau tên tệp chúng ta có thể thấy tên của đích mà liên kết trỏ tới, tức tệp `target.txt`.

Hãy lưu ý rằng trên danh sách tệp và thư mục, bản thân các liên kết mềm luôn hiển thị các quyền `rwx` cho người dùng, nhóm và những người khác. Nhưng trên thực tế, quyền truy cập đối với chúng cũng sẽ giống như quyền đối với đích.

Di chuyển và xóa các Liên kết Tượng trưng

Giống như các liên kết cứng, các liên kết tượng trưng có thể được gỡ bỏ bằng cách sử dụng `rm` và được di chuyển xung quanh hoặc đổi tên bằng cách sử dụng `mv`. Tuy nhiên, ta cần phải đặc biệt cẩn thận khi tạo chúng để tránh "phá vỡ" liên kết nếu nó bị di chuyển khỏi vị trí ban đầu.

Khi tạo các liên kết tượng trưng, chúng ta cần lưu ý rằng trừ khi một đường dẫn được chỉ định đầy đủ, vị trí của đích sẽ được hiểu *tương quan* với vị trí của liên kết. Điều này có thể tạo ra sự cố nếu liên kết hoặc tệp mà nó trỏ tới bị di chuyển. Ta có thể hiểu điều này một cách dễ dàng hơn với một ví dụ. Giả sử rằng chúng ta có một tệp có tên là `original.txt` trong thư mục hiện tại và chúng ta muốn tạo một liên kết tượng trưng tới tệp đó có tên là `softlink`. Ta có thể sử dụng:

```
$ ln -s original.txt softlink
```

Và có vẻ như tất cả đều ổn thoả. Hãy kiểm tra với `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
```

Hãy xem cách liên kết được tạo: `softlink` trỏ tới (→) `original.txt`. Tuy nhiên, hãy xem điều gì sẽ xảy ra nếu chúng ta di chuyển liên kết đến thư mục mẹ và cố gắng hiển thị nội dung của nó bằng lệnh `less`:

```
$ mv softlink ../
$ less ../softlink
../softlink: No such file or directory
```

Vì đường dẫn đến `original.txt` không được chỉ định nên hệ thống đã giả định rằng nó nằm trong cùng thư mục với liên kết. Khi điều này không còn đúng nữa, liên kết sẽ ngừng hoạt động.

Cách để ngăn chặn điều này xảy ra là luôn chỉ định đường dẫn đầy đủ đến đích khi tạo liên kết

```
$ ln -s /home/carol/Documents/original.txt softlink
```

Bằng cách này, bất kể ta di chuyển liên kết đến đâu thì nó cũngvẫn sẽ hoạt động vì nó trỏ đến vị trí tuyệt đối của đích. Hãy cùng kiểm tra với `ls`:

```
$ ls -lh
total 112K
lrwxrwxrwx 1 carol carol   40 Jun  7 19:34 softlink -> /home/carol/Documents/original.txt
```

Bài tập Hướng dẫn

1. Tham số cho `chmod` ở chế độ *tượng trưng* để gắn bit dính trên một thư mục là gì?

2. Hãy tưởng tượng có một tệp có tên `document.txt` trong thư mục `/home/carol/Documents`. Lệnh để tạo một liên kết tượng trưng đến nó có tên `text.txt` trong thư mục hiện tại là gì?

3. Hãy giải thích sự khác biệt giữa liên kết cứng tới một tệp và bản sao của tệp này.

Bài tập Mở rộng

1. Hãy tưởng tượng rằng bên trong một thư mục, bạn tạo một tệp có tên là `recipes.txt`. Bên trong thư mục này, bạn cũng sẽ tạo một liên kết cứng tới tệp này có tên là `receitas.txt` và một liên kết tượng trưng (hoặc liên kết *mềm*) tới tệp có tên là `rezepte.txt`.

   ```
   $ touch recipes.txt
   $ ln recipes.txt receitas.txt
   $ ln -s recipes.txt rezepte.txt
   ```

   Nội dung của thư mục sẽ xuất hiện như sau:

   ```
   $ 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 0K jun 17 17:25 recipes.txt
   5388837 lrwxrwxrwx 1 carol carol 12 jun 17 17:25 rezepte.txt -> receitas.txt
   ```

   Hãy nhớ rằng, với vai trò là một liên kết cứng, `receitas.txt` sẽ trỏ đến cùng một nút định danh giống như `recipes.txt`. Điều gì sẽ xảy ra với liên kết mềm `rezepte.txt` nếu như tên `receitas.txt` bị xóa? Tại sao?

2. Hãy tưởng tượng bạn có một ổ đĩa flash được cắm vào hệ thống của mình và được gắn kết trên `/media/youruser/FlashA`. Bạn muốn tạo trong thư mục chính của mình một liên kết có tên là `schematics.pdf` và trỏ tới tệp `esquema.pdf` trong thư mục gốc của ổ đĩa flash. Vì vậy, bạn đã gõ lệnh:

   ```
   $ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
   ```

   Chuyện gì sẽ xảy ra? Tại sao?

3. Hãy xem đầu ra sau của `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
```

- Có bao nhiêu liên kết trỏ đến tệp `document.txt`?
- Chúng là liên kết mềm hay cứng?
- Bạn nên truyền tham số nào cho `ls` để xem mỗi tệp ở trong nút định danh nào?

4. Hãy tưởng tượng trong thư mục `~/Documents` của bạn có một tệp tên là `clients.txt` có chứa tên của các khách hàng và một thư mục khác có tên là `somedir`. Bên trong đó có một tệp *khác cũng* có tên là `clients.txt` với những cái tên khác. Để sao chép cấu trúc này, hãy sử dụng các lệnh sau.

```
$ cd ~/Documents
$ echo "John, Michael, Bob" > clients.txt
$ mkdir somedir
$ echo "Bill, Luke, Karl" > somedir/clients.txt
```

Sau đó, bạn tạo một liên kết bên trong `somedir` có tên `partners.txt` trỏ đến tệp này bằng các lệnh:

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

Vì vậy, cấu trúc thư mục sẽ là:

```
Documents
|-- clients.txt
`-- somedir
  |-- clients.txt
  `-- partners.txt -> clients.txt
```

Bây giờ, bạn di chuyển `partners.txt` từ `somedir` sang `~/Documents` và liệt kê nội dung của nó.

```
$ cd ~/Documents/
```

```
$ mv somedir/partners.txt .
$ less partners.txt
```

Liệu liên kết có còn hoạt động hay không? Nếu có, tệp nào sẽ có nội dung được liệt kê? Tại sao?

5. Hãy xem các tệp sau:

```
-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
```

Quyền truy cập đối với `partners.txt` là gì? Tại sao?

Tóm tắt

Trong bài học này, chúng ta đã học về:

- Liên kết là gì.
- Sự khác biệt giữa liên kết *tượng trưng* và liên kết *cứng*.
- Cách tạo liên kết.
- Cách di chuyển, đổi tên hoặc loại bỏ các liên kết này.

Các lệnh sau đã được thảo luận trong bài học này:

- `ln`: Lệnh "link". Bản thân lệnh này sẽ tạo ra một liên kết cứng. Với khóa chuyển `-s`, một liên kết *tượng trưng* hoặc *mềm* có thể được tạo. Hãy nhớ rằng các liên kết cứng chỉ có thể nằm trên cùng một phân vùng và hệ thống tệp, còn các liên kết tượng trưng có thể đi qua các phân vùng và hệ thống tệp (thậm chí cả bộ lưu trữ nối với mạng).
- Tham số `-i` của `ls` cho phép người dùng xem số nút định danh của một tệp.

Đáp án Bài tập Hướng dẫn

1. Tham số cho `chmod` ở chế độ *tượng trưng* để gắn bit dính trên một thư mục là gì?

   Ký hiệu cho bit dính ở chế độ tượng trưng là `t`. Vì chúng ta muốn kích hoạt (thêm) quyền này vào thư mục nên tham số phải là `+t`.

2. Hãy tưởng tượng có một tệp có tên `document.txt` trong thư mục `/home/carol/Documents`. Lệnh để tạo một liên kết tượng trưng đến nó có tên `text.txt` trong thư mục hiện tại là gì?

   `ln -s` là lệnh tạo liên kết tượng trưng. Vì chúng ta phải chỉ định đường dẫn đầy đủ đến tệp đang liên kết tới nên lệnh sẽ là:

   ```
   $ ln -s /home/carol/Documents/document.txt text.txt
   ```

3. Hãy giải thích sự khác biệt giữa liên kết cứng tới một tệp và bản sao của tệp này.

   Liên kết cứng chỉ là tên gọi khác của một tệp. Mặc dù nó trông giống như một bản sao của tệp gốc nhưng xét trên mọi mục đích thì cả liên kết và tệp gốc đều giống nhau vì chúng trỏ đến cùng một dữ liệu trên đĩa. Những thay đổi được thực hiện trên nội dung của liên kết sẽ được phản ánh trên tệp gốc và ngược lại. Bản sao là một thực thể hoàn toàn độc lập, chiếm một vị trí khác ở trên đĩa. Những thay đổi trên bản sao sẽ không được phản ánh trên tệp gốc và ngược lại.

Đáp án Bài tập Mở rộng

1. Hãy tưởng tượng rằng bên trong một thư mục, bạn tạo một tệp có tên là `recipes.txt`. Bên trong thư mục này, bạn cũng sẽ tạo một liên kết cứng tới tệp này có tên là `receitas.txt` và một liên kết tượng trưng (hoặc liên kết *mềm*) tới tệp có tên là `rezepte.txt`.

   ```
   $ touch recipes.txt
   $ ln recipes.txt receitas.txt
   $ ln -s receitas.txt rezepte.txt
   ```

   Nội dung của thư mục sẽ xuất hiện như sau:

   ```
   $ 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
   ```

   Hãy nhớ rằng, với vai trò là một liên kết cứng, `receitas.txt` sẽ trỏ đến cùng một nút định danh giống như `recipes.txt`. Điều gì sẽ xảy ra với liên kết mềm `rezepte.txt` nếu như tên `receitas.txt` bị xóa? Tại sao?

   Liên kết mềm `rezepte.txt` sẽ ngừng hoạt động. Điều này là do các liên kết mềm trỏ đến tên chứ không phải nút định danh, và tên `receitas.txt` không còn tồn tại ngay cả khi dữ liệu vẫn còn trên đĩa dưới tên `recipes.txt`.

2. Hãy tưởng tượng bạn có một ổ đĩa flash được cắm vào hệ thống của mình và được gắn kết trên `/media/youruser/FlashA`. Bạn muốn tạo trong thư mục chính của mình một liên kết có tên là `schematics.pdf` và trỏ tới tệp `esquema.pdf` trong thư mục gốc của ổ đĩa flash. Vì vậy, bạn đã gõ lệnh:

   ```
   $ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
   ```

   Chuyện gì sẽ xảy ra? Tại sao?

   Lệnh sẽ thất bại. Thông báo lỗi sẽ là `Invalid cross-device link` (Liên kết liên thiết bị không hợp lệ) và nó đã làm rõ lý do: các liên kết cứng không thể trỏ đến mục tiêu trong một phân vùng hoặc thiết bị khác. Cách duy nhất để tạo liên kết như thế này là sử dụng liên kết *tượng trưng* hoặc liên kết *mềm* với tham số `-s` vào `ln`.

3. Hãy xem đầu ra sau của `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
```

   - Có bao nhiêu liên kết trỏ đến tệp `document.txt`?

     Mỗi tệp đều bắt đầu với số lượng liên kết là `1`. Vì số liên kết cho tệp là `4` nên ở đây sẽ có ba liên kết trỏ đến tệp đó.

   - Chúng là liên kết mềm hay cứng?

     Chúng là các liên kết cứng vì các liên kết mềm không làm tăng số lượng liên kết của tệp.

   - Bạn nên truyền tham số nào cho `ls` để xem mỗi tệp ở trong nút định danh nào?

     Tham số là `-i`. Nút định danh sẽ được hiển thị dưới dạng cột đầu tiên trong đầu ra của `ls` như bên dưới đây:

```
$ ls -lahi
total 3,1M
5388773 drwxr-xr-x 2 rigues rigues 4,0K jun 17 17:27 .
5245554 drwxr-xr-x 5 rigues rigues 4,0K jun 17 17:29 ..
5388840 -rw-rw-r-- 1 rigues rigues 2,8M jun 17 15:45 compressed.zip
5388833 -rw-r--r-- 4 rigues rigues 77K jun 17 17:25 document.txt
5388837 -rw-rw-r-- 1 rigues rigues 216K jun 17 17:25 image.png
5388833 -rw-r--r-- 4 rigues rigues 77K jun 17 17:25 text.txt
```

4. Hãy tưởng tượng trong thư mục `~/Documents` của bạn có một tệp tên là `clients.txt` nhưng có chứa các tên khách hàng khác và một thư mục có tên là `somedir`. Bên trong đó có một tệp *khác cũng* có tên là `clients.txt` với những cái tên khác. Để sao chép cấu trúc này, hãy sử dụng các lệnh sau.

```
$ cd ~/Documents
$ echo "John, Michael, Bob" > clients.txt
```

```
$ mkdir somedir
$ echo "Bill, Luke, Karl" > somedir/clients.txt
```

Sau đó, bạn tạo một liên kết bên trong `somedir` có tên `partners.txt` trỏ đến tệp này bằng các lệnh:

```
$ cd somedir/
$ ln -s clients.txt partners.txt
```

Vì vậy, cấu trúc thư mục sẽ là:

```
Documents
|-- clients.txt
`-- somedir
  |-- clients.txt
  `-- partners.txt -> clients.txt
```

Bây giờ, bạn di chuyển `partners.txt` từ `somedir` sang `~/Documents` và liệt kê nội dung của nó.

```
$ cd ~/Documents/
$ mv somedir/partners.txt .
$ less partners.txt
```

Liệu liên kết có còn hoạt động hay không? Nếu có, tệp nào sẽ có nội dung được liệt kê? Tại sao?

Đây là một câu hỏi "bẫy"; nhưng liên kết vẫn sẽ hoạt động và tệp được liệt kê sẽ là tệp trong `~/Documents` có chứa các tên `John`, `Michael`, `Bob`.

Hãy nhớ rằng vì bạn không chỉ định đường dẫn đầy đủ đến đích `clients.txt` khi tạo liên kết mềm `partners.txt` nên vị trí đích sẽ được hiểu là tương quan với vị trí của liên kết, trong trường hợp này là thư mục hiện tại.

Khi liên kết được di chuyển từ `~/Documents/somedir` sang `~/Documents`, liên kết sẽ ngừng hoạt động vì đích không còn nằm trong cùng thư mục với liên kết. Tuy nhiên, tình cờ là có một tệp có tên `clients.txt` trên `~/Documents`; vì vậy, liên kết sẽ trỏ đến tệp này thay vì đích ban đầu bên trong `~/somedir`.

Để tránh điều này, hãy luôn chỉ định đường dẫn đầy đủ đến đích khi tạo liên kết tượng trưng.

5. Hãy xem các tệp sau:

```
-rw-r--r-- 1 rigues rigues 19 Jun 24 11:12 clients.txt
lrwxrwxrwx 1 rigues rigues 11 Jun 24 11:13 partners.txt -> clients.txt
```

Quyền truy cập đối với `partners.txt` là gì? Tại sao?

Quyền truy cập cho `partners.txt` là `rw-r—r--` vì các liên kết luôn kế thừa các quyền truy cập giống như đích.

104.7 Tìm Tệp hệ thống và đặt Tệp vào đúng vị trí

Tham khảo các mục tiêu LPI

LPIC-1 version 5.0, Exam 101, Objective 104.7

Khối lượng

2

Các lĩnh vực kiến thức chính

- Hiểu vị trí chính xác của các tệp trong FHS.
- Tìm tệp và lệnh trên hệ thống Linux.
- Biết vị trí và mục đích của các tệp và thư mục quan trọng như được xác định trong FHS.

Sau đây là danh sách một phần các tệp, thuật ngữ và tiện ích được sử dụng

- `find`
- `locate`
- `updatedb`
- `whereis`
- `which`
- `type`
- `/etc/updatedb.conf`

Linux Professional Institute

104.7 Bài 1

Chứng chỉ:	LPIC-1
Phiên bản:	5.0
Chủ đề:	104 Thiết bị, Hệ thống tệp Linux, Tiêu chuẩn phân cấp hệ thống Tệp
Mục tiêu:	104.7 Tìm Tệp hệ thống và đặt Tệp vào đúng vị trí
Bài:	1 trên 1

Giới thiệu

Các bản phân phối của Linux rất đa dạng, nhưng có một điều mà hầu hết chúng đều tuân theo là *Tiêu chuẩn Phân cấp Hệ thống Tệp* (FHS) xác định "bố cục tiêu chuẩn" cho hệ thống tệp, giúp việc tương tác và quản trị hệ thống trở nên dễ dàng hơn rất nhiều. Trong bài học này, chúng ta sẽ tìm hiểu thêm về tiêu chuẩn này và cách tìm tệp trên hệ thống Linux.

Tiêu chuẩn Phân cấp Hệ thống Tệp

Tiêu chuẩn Phân cấp Hệ thống Tệp (FHS) là một nỗ lực của Hiệp hội Linux (Linux Foundation) nhằm chuẩn hóa cấu trúc thư mục và nội dung thư mục trên hệ thống Linux. Việc tuân thủ tiêu chuẩn không phải là bắt buộc, nhưng hầu hết các bản phân phối đều tuân theo tiêu chuẩn này.

NOTE Những người quan tâm đến chi tiết về tổ chức hệ thống tệp có thể đọc đặc tả FHS 3.0 có sẵn ở nhiều định dạng tại: http://refspecs.linuxfoundation.org/fhs.shtml

Theo tiêu chuẩn, cấu trúc thư mục cơ bản như sau:

`/`

Đây là thư mục gốc, thư mục trên cùng trong hệ thống phân cấp. Mọi thư mục khác đều nằm bên trong nó. Một hệ thống tệp thường được so sánh với một "cây". Vì thế, đây sẽ là "thân cây" mà tất cả các nhánh được kết nối tới.

`/bin`

Các tệp nhị phân cần thiết, có sẵn cho tất cả người dùng.

`/boot`

Các tệp cần thiết cho quá trình khởi động, bao gồm Ổ đĩa RAM ban đầu (initrd) và bản thân nhân Linux.

`/dev`

Tệp thiết bị. Đây có thể là các thiết bị vật lý được kết nối với hệ thống (ví dụ như `/dev/sda` sẽ là đĩa SCSI hoặc đĩa SATA đầu tiên) hoặc các thiết bị ảo do hạt nhân cung cấp.

`/etc`

Tệp cấu hình dành riêng cho máy chủ. Các chương trình có thể tạo thư mục con trong `/etc` để lưu trữ nhiều tệp cấu hình nếu cần.

`/home`

Mỗi người dùng trong hệ thống đều có một thư mục "chính" để lưu trữ các tệp và sở thích cá nhân, và hầu hết chúng đều nằm trong `/home`. Thông thường, thư mục chính sẽ giống với tên người dùng. Vì vậy, người dùng John sẽ có thư mục của mình dưới `/home/john`. Các trường hợp ngoại lệ là siêu người dùng (root) có thư mục riêng (`/root`) và một số người dùng hệ thống.

`/lib`

Cần có các thư viện chia sẻ để khởi động hệ điều hành và chạy các tệp nhị phân trong `/bin` và `/sbin`.

`/media`

Phương tiện di động có thể gắn kết với người dùng như ổ đĩa flash, đầu đọc CD và DVD-ROM, đĩa mềm, thẻ nhớ và ổ đĩa cứng ngoại vị đều được gắn ở đây.

`/mnt`

Điểm gắn kết cho các hệ thống tệp được gắn tạm thời.

`/opt`

Các gói phần mềm ứng dụng.

`/root`

Thư mục chính cho siêu người dùng (root).

`/run`

Dữ liệu biến thời gian chạy.

`/sbin`

Tệp nhị phân hệ thống.

`/srv`

Dữ liệu do hệ thống cung cấp. Ví dụ: các trang do máy chủ web cung cấp có thể được lưu trữ trong `/srv/www`.

`/tmp`

Các tệp tạm thời.

`/usr`

Dữ liệu người dùng chỉ đọc, bao gồm dữ liệu cần thiết cho một số tiện ích và ứng dụng phụ.

`/proc`

Hệ thống tệp ảo chứa dữ liệu liên quan đến các tiến trình đang chạy.

`/var`

Dữ liệu biến được ghi trong quá trình vận hành hệ thống, bao gồm hàng đợi in, dữ liệu nhật ký, hộp thư, tệp tạm thời, bộ nhớ đệm của trình duyệt, v.v.

Hãy nhớ rằng một trong số các thư mục trên (như `/etc`, `/usr` và `/var`) sẽ chứa toàn bộ hệ thống phân cấp của các thư mục con bên dưới chúng.

Tệp tạm thời

Tệp tạm thời là các tệp được chương trình sử dụng để lưu trữ dữ liệu chỉ cần thiết trong một khoảng thời gian ngắn. Đây có thể là dữ liệu của các tiến trình đang chạy, nhật ký sự cố, tệp scratch từ quá trình lưu tự động, tệp trung gian được sử dụng trong quá trình chuyển đổi tệp, tệp bộ nhớ đệm, v.v.

Vị trí của Tệp tạm thời

Phiên bản 3.0 của *Tiêu chuẩn Phân cấp Hệ thống Tệp* (FHS) có xác định vị trí tiêu chuẩn cho các tệp tạm thời trên hệ thống Linux. Mỗi vị trí có một mục đích và hành vi khác nhau và các nhà phát triển nên tuân theo các quy ước do FHS đặt ra khi ghi dữ liệu tạm thời vào đĩa.

`/tmp`

Theo FHS, các chương trình không nên giả định rằng các tệp được ghi ở đây sẽ được giữ nguyên giữa các lần gọi chương trình. Khuyến nghị cho người dùng là nên xóa thư mục này (xóa tất cả các tệp) trong quá trình khởi động hệ thống (mặc dù điều này không phải là bắt buộc).

`/var/tmp`

Một vị trí khác dành cho các tệp tạm thời, nhưng vị trí này *không nên bị xóa sạch* trong quá trình khởi động hệ thống. Các tệp được lưu trữ ở đây thường vẫn sẽ tiếp tục tồn tại giữa các lần khởi động lại.

`/run`

Thư mục này chứa dữ liệu biến thời gian chạy được sử dụng bởi các tiến trình đang chạy (chẳng hạn như các tệp định danh tiến trình - `.pid`). Các chương trình cần nhiều hơn một tệp thời gian chạy nơi có thể tạo các thư mục con tại đây. Vị trí này *phải được xóa sạch* trong quá trình khởi động hệ thống. `/var/run` từng được sử dụng để phục vụ cho mục đích này và trên một số hệ thống `/var/run` có thể là một liên kết tượng trưng đến `/run`.

Hãy lưu ý rằng không có gì ngăn cản một chương trình tạo các tệp tạm thời ở những nơi khác trên hệ thống, nhưng người dùng nên tôn trọng các quy ước do FHS đặt ra.

Tìm Tệp

Để tìm kiếm tệp trên hệ thống Linux, chúng ta có thể sử dụng lệnh `find`. Đây là một công cụ rất mạnh mẽ, có đầy đủ các tham số phù hợp với hành vi của nó và có thể sửa đổi đầu ra chính xác theo nhu cầu của người dùng.

Để bắt đầu, `find` cần có hai đối số: điểm bắt đầu và nội dung cần tìm. Ví dụ: để tìm kiếm tất cả các tệp trong thư mục hiện tại (và các thư mục con) có tên kết thúc bằng `.jpg`, ta có thể sử dụng:

```
$ find . -name '*.jpg'
./pixel_3a_seethrough_1.jpg
./Mate3.jpg
./Expert.jpg
./Pentaro.jpg
./Mate1.jpg
./Mate2.jpg
./Sala.jpg
./Hotbit.jpg
```

Vì `*` là ký tự đại diện cho “bất cứ một ký tự nào”, lệnh này sẽ trùng khớp với bất kỳ tệp nào có bốn

ký tự cuối cùng trong tên là `.jpg` dù ký tự đứng trước nó là gì. Tuy nhiên, hãy xem điều gì sẽ xảy ra nếu một `*` khác được thêm vào *phần cuối* của mẫu:

```
$ 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
```

Tệp `Pentaro.jpg.zip` (được in đậm ở trên) không có trong danh sách trước đó vì ngay cả khi nó có chứa `.jpg` trên tên của nó, nó không khớp với mẫu vì có các ký tự phụ đứng sau nó. Mẫu mới có nghĩa là "bất cứ thứ gì `.jpg` bất cứ thứ gì", vì vậy nên nó mới có thể khớp tệp này.

TIP Hãy nhớ rằng tham số `-name` có phân biệt chữ hoa và chữ thường. Nếu muốn thực hiện tìm kiếm không phân biệt chữ hoa chữ thường, hãy sử dụng `-iname`.

Biểu thức `*.jpg` phải được đặt bên trong dấu trích dẫn đơn để tránh vỏ tự diễn giải mẫu. Hãy xem thử trường hợp không có dấu trích dẫn kép và xem điều gì sẽ xảy ra.

Theo mặc định, `find` sẽ bắt đầu ở điểm bắt đầu và đi xuống bất kỳ một thư mục con nào (và các thư mục con của các thư mục con đó) được tìm thấy. Ta có thể hạn chế hành vi này bằng tham số `-maxdeep N`, trong đó, `N` là số cấp độ tối đa.

Để chỉ tìm kiếm trong thư mục hiện tại, ta có thể sử dụng `-maxdeep 1`. Giả sử chúng ta có cấu trúc thư mục sau:

```
directory
├── clients.txt
├── partners.txt -> clients.txt
└── somedir
    ├── anotherdir
    └── clients.txt
```

Để tìm kiếm bên trong `somedir`, ta cần sử dụng `-maxdeep 2` (thư mục hiện tại xuống 1 cấp). Để tìm kiếm bên trong `anotherdir`, ta cần có `-maxdeep 3` (thư mục hiện tại xuống 2 cấp). Tham số `-mindeep N` hoạt động theo cách ngược lại, tức là chỉ tìm kiếm trong các thư mục từ *ít nhất* từ cấp

`N` trở xuống.

Tham số `-mount` có thể được sử dụng để tránh `find` đi xuống bên trong hệ thống tệp đã được gắn kết. Ta cũng có thể hạn chế tìm kiếm ở các loại hệ thống tệp cụ thể bằng cách sử dụng tham số `-fstype`. Vì vậy, `find /mnt -fstype exfat -iname "*report*"` sẽ chỉ tìm kiếm bên trong các hệ thống tệp exFAT được gắn trong `/mnt`.

Tìm kiếm Thuộc tính

Chúng ta có thể sử dụng các tham số bên dưới để tìm kiếm các tệp có thuộc tính cụ thể (chẳng hạn như các tệp mà người dùng có thể ghi, có một bộ quyền cụ thể hoặc có một kích thước nhất định):

`-user USERNAME`
: So khớp các tệp thuộc sở hữu của người dùng `USERNAME`.

`-group GROUPNAME`
: So khớp các tệp thuộc sở hữu của nhóm `GROUPNAME`.

`-readable`
: So khớp các tệp mà người dùng hiện tại có thể đọc được.

`-writable`
: So khớp các tệp mà người dùng hiện tại có thể ghi.

`-executable`
: So khớp các tệp mà người dùng hiện tại có thể thực thi được. Đối với thư mục, tham số này sẽ khớp với bất kỳ thư mục nào mà người dùng có thể truy nhập (quyền `x`).

`-perm NNNN`
: Tham số này sẽ khớp với bất kỳ tệp nào có chính xác quyền `NNNN`. Ví dụ: `-perm 0664` sẽ khớp với bất kỳ tệp nào mà người dùng và nhóm có thể đọc và ghi và những tệp mà người khác có thể đọc (hoặc `rw-rw-r--`).

Chúng ta có thể thêm `-` vào trước `NNNN` để kiểm tra các tệp có *ít nhất* quyền được chỉ định. Ví dụ: `-perm -644` sẽ khớp với các tệp có ít nhất quyền `644` (`rw-r—r--`) - bao gồm một tệp có `664` (`rw-rw-r--`) hoặc thậm chí là `775` (`rwxrwx-r-x`).

`-empty`
: Sẽ khớp với các tệp và thư mục trống.

`-size N`

Sẽ khớp với bất kỳ tệp nào có kích thước `N`, trong đó, `N` theo mặc định là một số khối 512 byte. Ta có thể thêm hậu tố vào `N` cho các đơn vị khác: `Nc` sẽ tính kích thước tính bằng byte, `Nk` tính bằng kibibytes (KiB, bội số của 1024 byte), `NM` tính bằng mebibytes (MiB, bội số của 1024 * 1024) và `NG` cho gibibyte (GiB, bội số của 1024 * 1024 * 1024).

Một lần nữa, chúng ta có thể thêm tiền tố `+` hoặc `-` (ở đây có nghĩa là *lớn hơn* và *nhỏ hơn*) để tìm kiếm kích thước tương đối. Ví dụ: `-size -10M` sẽ khớp với bất kỳ tệp nào có kích thước nhỏ hơn 10 MiB.

Ví dụ: để tìm kiếm các tệp trong thư mục chính có chứa mẫu `report` không phân biệt chữ hoa chữ thường trong bất kỳ phần nào của tên, có quyền `0644`, đã được truy cập 10 ngày trước và có kích thước ít nhất là 1 Mib, chúng ta có thể sử dụng:

```
$ find ~ -iname "*report*" -perm 0644 -atime 10 -size +1M
```

Tìm kiếm theo Thời gian

Bên cạnh việc tìm kiếm các thuộc tính, chúng ta cũng có thể thực hiện tìm kiếm theo thời gian để tìm các tệp đã được truy cập, các thuộc tính của chúng đã thay đổi hoặc đã được sửa đổi trong một khoảng thời gian cụ thể. Các tham số sẽ là:

`-amin N`, **`-cmin N`**, **`-mmin N`**

Các tham số này sẽ khớp với các tệp đã được truy cập và có thuộc tính đã thay đổi hoặc đã được sửa đổi (tương ứng) vào `N` phút trước.

`-atime N`, **`-ctime N`**, **`-mtime N`**

Các tham số này sẽ khớp với các tệp đã được truy cập, có thuộc tính đã thay đổi hoặc đã được sửa đổi vào `N*24` giờ trước.

Đối với `-cmin N` và `-ctime N`, bất kỳ thay đổi thuộc tính nào cũng sẽ khớp (bao gồm thay đổi về quyền, đọc hoặc ghi vào tệp). Điều này làm cho các tham số này trở nên đặc biệt mạnh mẽ vì trên thực tế, bất kỳ thao tác nào liên quan đến tệp cũng sẽ được khớp.

Ví dụ sau sẽ khớp với bất kỳ tệp nào trong thư mục hiện tại đã được sửa đổi chưa đầy 24 giờ trước và lớn hơn 100 MiB:

```
$ find . -mtime -1 -size +100M
```

Sử dụng `locate` và `updatedb`

`locate` và `updatedb` là các lệnh có thể được sử dụng để nhanh chóng tìm thấy một tệp khớp với mẫu đã cho trên hệ thống Linux. Nhưng không giống như `find`, `locate` sẽ không tìm kiếm mẫu trong hệ thống tệp. Thay vào đó, nó sẽ tìm kiếm mẫu đó trên cơ sở dữ liệu được xây dựng bằng cách chạy lệnh `updatedb`. Điều này mang lại cho chúng ta kết quả rất nhanh nhưng có thể sẽ không chính xác, tùy thuộc vào thời điểm cơ sở dữ liệu được cập nhật lần cuối cùng.

Cách đơn giản nhất để sử dụng `locate` là chỉ cần cung cấp cho nó một mẫu tìm kiếm. Ví dụ: để tìm mọi hình ảnh JPEG trên hệ thống, chúng ta sẽ sử dụng `locate jpg`. Danh sách kết quả có thể sẽ khá dài và nó sẽ có dạng như sau:

```
$ 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
```

Khi được yêu cầu về mẫu `jpg`, `locate` sẽ hiển thị bất kỳ nội dung nào có chứa mẫu này bất kể phần đứng trước hay sau nó là gì. Chúng ta có thể xem ví dụ về điều này trong tệp `jpg_specs.doc` trong danh sách ở trên: nó có chứa mẫu nhưng phần mở rộng lại không phải là `jpg`.

TIP Hãy nhớ rằng với `locate`, bạn đang khớp các mẫu chứ không phải phần mở rộng tệp.

Theo mặc định, mẫu có phân biệt chữ hoa chữ thường. Điều này có nghĩa là các tệp chứa `.JPG` sẽ không được hiển thị do mẫu của chúng ta đang ở dạng chữ thường. Để tránh điều này, hãy truyền tham số `-i` cho `locate`. Nếu lặp lại ví dụ trước:

```
$ 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
```

Hãy lưu ý rằng tệp `Mate1_old.JPG` được in đậm ở trên không hề có trong danh sách trước đó.

Ta có thể truyền nhiều mẫu cho `locate` và chỉ cần phân tách chúng bằng dấu cách. Ví dụ bên dưới sẽ thực hiện tìm kiếm không phân biệt chữ hoa chữ thường đối với bất kỳ tệp nào khớp với mẫu `zip` và `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
```

Khi sử dụng nhiều mẫu, ta có thể yêu cầu locate chỉ hiển thị các tệp khớp với *tất cả các mẫu* trong số đó. Điều này được thực hiện với tùy chọn `-A`. Ví dụ sau đây sẽ hiển thị bất kỳ tệp nào khớp với các mẫu `.jpg` *và* `.zip`:

```
$ locate -A .jpg .zip
/home/carol/Downloads/Pentaro.jpg.zip
```

Nếu muốn đếm số lượng tệp khớp với một mẫu nhất định thay vì hiển thị đường dẫn đầy đủ của chúng, ta có thể sử dụng tùy chọn `-c`. Ví dụ: để đếm số lượng tệp `.jpg` trên một hệ thống:

```
$ locate -c .jpg
1174
```

Có một vấn đề với `locate` là nó sẽ chỉ hiển thị các mục có trong cơ sở dữ liệu được tạo bởi `updateb` (nằm trong `/var/lib/mlocate.db`). Nếu cơ sở dữ liệu đã lỗi thời, kết quả đầu ra có thể hiển thị các tệp đã bị xóa kể từ lần cập nhật gần đây nhất. Một cách để tránh điều này là thêm

tham số `-e`. Tham số này sẽ kiểm tra xem liệu tệp có còn tồn tại hay không trước khi hiển thị nó trên đầu ra.

Tất nhiên, điều này sẽ không giải quyết được vấn đề các tệp được tạo *sau* bản cập nhật cơ sở dữ liệu cuối cùng không hiển thị. Để làm được điều này, chúng ta sẽ phải cập nhật cơ sở dữ liệu bằng lệnh `updateb`. Thời gian thực hiện tiến trình này tùy thuộc vào số lượng tệp trên đĩa của người dùng.

Kiểm soát hành vi của `updatedb`

Hành vi của `updatedb` có thể được kiểm soát bởi tệp `/etc/updatedb.conf`. Đây là một tệp văn bản mà ở trong đó, mỗi dòng sẽ điều khiển một biến. Các dòng trống sẽ bị bỏ qua và các dòng bắt đầu bằng ký tự # sẽ được coi là chú thích.

`PRUNEFS=`

Bất kỳ loại hệ thống tệp nào được chỉ định sau tham số này sẽ không được `updatedb` quét. Danh sách các loại phải được phân tách bằng dấu cách và bản thân các loại sẽ không phân biệt chữ hoa chữ thường. Vì thế nên `NFS` và `nfs` sẽ tương đương với nhau.

`PRUNENAMES=`

Đây là danh sách các tên thư mục được phân tách bằng dấu cách mà `updatedb` không quét.

`PRUNEPATHS=`

Đây là danh sách các tên đường dẫn nên được `updatedb` bỏ qua. Tên đường dẫn phải được phân tách bằng dấu cách và được chỉ định giống như cách chúng được hiển thị bởi `updatedb` (ví dụ: `/var/spool /media`)

`PRUNE_BIND_MOUNTS=`

Đây là một biến `yes` hoặc `no` đơn giản. Nếu được đặt thành `yes`, các liên kết gắn kết (các thư mục được gắn ở nơi khác bằng lệnh `mount --bind`) sẽ bị bỏ qua.

Tìm Tệp nhị phân, Trang hướng dẫn và Mã nguồn

`which` là một lệnh rất hữu ích được sử dụng để hiển thị đường dẫn đầy đủ đến tệp thực thi. Ví dụ: nếu muốn định vị tệp thực thi cho `bash`, ta có thể sử dụng:

```
$ which bash
/usr/bin/bash
```

Nếu tùy chọn `-a` được thêm vào, lệnh sẽ hiển thị tất cả các tên đường dẫn khớp với tệp thực thi.

Hãy quan sát sự khác biệt:

```
$ which mkfs.ext3
/usr/sbin/mkfs.ext3

$ which -a mkfs.ext3
/usr/sbin/mkfs.ext3
/sbin/mkfs.ext3
```

TIP Để tìm thư mục nào đang có trong `PATH`, hãy sử dụng lệnh `echo $PATH`. Lệnh này sẽ in (`echo`) nội dung của biến `PATH` (`$PATH`) trên cửa sổ dòng lệnh của bạn.

`type` là một lệnh tương tự sẽ hiển thị thông tin về tệp nhị phân, bao gồm vị trí và loại của nó. Ta chỉ cần sử dụng `type`, theo sau là tên lệnh:

```
$ type locate
locate is /usr/bin/locate
```

Tham số `-a` cũng hoạt động theo cách tương tự như trong `which` và sẽ hiển thị tất cả các tên đường dẫn khớp với tệp thực thi giống như sau:

```
$ type -a locate
locate is /usr/bin/locate
locate is /bin/locate
```

Và tham số `-t` sẽ hiển thị loại tệp của lệnh, có thể là `alias`, `keyword`, `function`, `buildin` hoặc `file`. Ví dụ:

```
$ type -t locate
file

$ type -t ll
alias

$ type -t type
type is a built-in shell command
```

Lệnh `whereis` sẽ linh hoạt hơn và ngoài ra, các tệp nhị phân cũng có thể được sử dụng để hiển thị vị trí của các trang hướng dẫn hoặc thậm chí là mã nguồn của một chương trình (nếu có trong hệ

thống). Ta chỉ cần gõ `whereis`, theo sau là tên tệp nhị phân:

```
$ whereis locate
locate: /usr/bin/locate /usr/share/man/man1/locate.1.gz
```

Các kết quả ở trên bao gồm các tệp nhị phân (`/usr/bin/locate`) và các trang hướng dẫn được nén (`/usr/share/man/man1/locate.1.gz`).

Ta có thể nhanh chóng lọc kết quả bằng cách sử dụng các khoá chuyển dòng lệnh như `-b` (giới hạn kết quả ở các tệp nhị phân), `-m` (giới hạn kết quả ở các trang hướng dẫn) hoặc `-s` (giới hạn kết quả ở mã nguồn). Lặp lại ví dụ trên, ta sẽ nhận được:

```
$ whereis -b locate
locate: /usr/bin/locate

$ whereis -m locate
locate: /usr/share/man/man1/locate.1.gz
```

Bài tập Hướng dẫn

1. Hãy tưởng tượng một chương trình cần tạo một tệp tạm thời dùng một lần và sẽ không bao giờ cần đến nó nữa sau khi đóng chương trình. Ta nên tạo tệp này ở trong thư mục nào?

2. Thư mục tạm thời nào *phải* bị xóa trong quá trình khởi động?

3. Bằng cách sử dụng `find`, hãy tìm kiếm chỉ trong thư mục hiện tại những tệp mà người dùng có thể ghi, đã được sửa đổi trong 10 ngày qua và lớn hơn 4 GiB.

4. Bằng cách sử dụng `locate`, hãy tìm bất kỳ tệp nào có chứa cả hai mẫu `report` và `updated` /`update`/`updating` trong tên của chúng.

5. Làm cách nào để có thể biết được trang hướng dẫn cho `ifconfig` được lưu trữ ở đâu?

6. Biến nào cần được thêm vào `/etc/updatedb.conf` để khiến `updatedb` bỏ qua hệ thống tệp `ntfs`?

7. Quản trị viên hệ thống muốn gắn kết một đĩa cục bộ (`/dev/sdc1`). Theo FHS, đĩa này nên được gắn vào thư mục nào?

Bài tập Mở rộng

1. Khi `locate` được sử dụng, kết quả sẽ được lấy từ cơ sở dữ liệu do `updatedb` tạo ra. Tuy nhiên, cơ sở dữ liệu này có thể đã lỗi thời và khiến cho `locate` hiển thị các tệp không còn tồn tại. Làm cách nào để có thể khiến `locate` chỉ hiển thị các tệp hiện có trên đầu ra của nó?

2. Hãy tìm bất kỳ tệp nào trên thư mục hiện tại hoặc các thư mục con từ hai cấp trở xuống (ngoại trừ các hệ thống tệp được gắn kết) có chứa mẫu `Status` hoặc `statute` trong tên của chúng.

3. Với giới hạn tìm kiếm trong các hệ thống tệp `ext4`, hãy tìm bất kỳ tệp nào trong `/mnt` có ít nhất quyền thực thi cho nhóm, có thể đọc được đối với người dùng hiện tại và có bất kỳ thuộc tính nào đã được thay đổi trong 2 giờ qua.

4. Hãy tìm các tệp trống được tạo cách đây hơn 30 ngày và xuống dưới ít nhất hai cấp so với thư mục hiện tại.

5. Giả sử người dùng `carol` và `john` là một phần của nhóm `mkt`. Hãy tìm trong thư mục chính của `john` bất kỳ tệp nào mà `carol` cũng có thể đọc được.

Tóm tắt

Trong bài học này, chúng ta đã tìm hiểu về cách tổ chức cơ bản của hệ thống tệp trên máy Linux theo như FHS cũng như cách tìm các tệp nhị phân và tệp theo tên hoặc thuộc tính. Các lệnh sau đã được thảo luận trong bài học này: `find`:: Một lệnh linh hoạt được sử dụng để tìm các tệp và thư mục dựa trên nhiều tiêu chí tìm kiếm khác nhau. `locate`:: Một tiện ích sử dụng một cơ sở dữ liệu cục bộ chứa vị trí của các tệp được lưu trữ cục bộ. `updatedb`:: Cập nhật cơ sở dữ liệu cục bộ được sử dụng bởi lệnh `locate`. `which`:: Hiển thị đường dẫn đầy đủ đến tệp thực thi. `whereis`:: Hiển thị vị trí của các trang thủ công, tệp nhị phân và mã nguồn trên hệ thống. `type`:: Hiển thị vị trí của tệp nhị phân và loại ứng dụng của nó (chẳng hạn như chương trình đã được cài đặt, chương trình Bash tích hợp sẵn và hơn thế nữa).

Đáp án Bài tập Hướng dẫn

1. Hãy tưởng tượng một chương trình cần tạo một tệp tạm thời dùng một lần và sẽ không bao giờ cần đến nó nữa sau khi đóng chương trình. Ta nên tạo tệp này chính xác trong thư mục nào?

   Vì không quan tâm đến tệp sau khi chương trình chạy xong nên thư mục chính xác sẽ là `/tmp`.

2. Thư mục tạm thời nào *phải* bị xóa trong quá trình khởi động?

   Thư mục là `/run` hoặc `/var/run` trên một số hệ thống.

3. Bằng cách sử dụng `find`, hãy tìm kiếm chỉ trong thư mục hiện tại những tệp mà người dùng có thể ghi, đã được sửa đổi trong 10 ngày qua và lớn hơn 4 GiB.

   Để làm điều này, bạn sẽ cần các tham số `-writable`, `-mtime` và `-size`:

   ```
   find . -writable -mtime -10 -size +4G
   ```

4. Bằng cách sử dụng `locate`, hãy tìm bất kỳ tệp nào có chứa cả hai mẫu `report` và `updated` /`update`/`updating` trong tên của chúng.

   Vì `locate` cần khớp với tất cả các mẫu, hãy sử dụng tùy chọn `-A`:

   ```
   locate -A "report" "updat"
   ```

5. Làm cách nào để có thể biết được trang hướng dẫn cho `ifconfig` được lưu trữ ở đâu?

   Sử dụng tham số `-m` cho `whereis`:

   ```
   whereis -m ifconfig
   ```

6. Biến nào cần được thêm vào `/etc/updatedb.conf` để khiến `updatedb` bỏ qua hệ thống tệp `ntfs`?

   Biến là `PRUNEFS=`, theo sau là loại hệ thống tệp: `PRUNEFS=ntfs`.

7. Quản trị viên hệ thống muốn gắn kết một đĩa cục bộ (`/dev/sdc1`). Theo FHS, đĩa này nên được gắn vào thư mục nào?

   Trong thực tế, đĩa có thể được gắn kết ở bất cứ đâu. Tuy nhiên, FHS khuyến nghị việc gắn kết tạm thời nên được thực hiện tại `/mnt`.

Đáp án Bài tập Mở rộng

1. Khi `locate` được sử dụng, kết quả sẽ được lấy từ cơ sở dữ liệu do `updatedb` tạo ra. Tuy nhiên, cơ sở dữ liệu này có thể đã lỗi thời và khiến cho `locate` hiển thị các tệp không còn tồn tại. Làm cách nào để có thể khiến `locate` chỉ hiển thị các tệp hiện có trên đầu ra của nó?

   Thêm tham số `-e` cho locate như trong `locate -e PATTERN`.

2. Hãy tìm bất kỳ tệp nào trên thư mục hiện tại hoặc các thư mục con từ hai cấp trở xuống (ngoại trừ các hệ thống tệp được gắn kết) có chứa mẫu `Status` hoặc `statute` trong tên của chúng.

   Hãy nhớ rằng đối với `-maxdeep`, bạn cũng phải tính cả thư mục hiện tại. Vì vậy, chúng ta cần có ba cấp độ (thư mục hiện tại cộng với 2 cấp độ giảm xuống):

   ```
   find . -maxdepth 3 -mount -iname "*statu*"
   ```

3. Với giới hạn tìm kiếm trong các hệ thống tệp `ext4`, hãy tìm bất kỳ tệp nào trong `/mnt` có ít nhất quyền thực thi cho nhóm, có thể đọc được đối với người dùng hiện tại và có bất kỳ thuộc tính nào đã được thay đổi trong 2 giờ qua.

   Sử dụng tham số `-fstype` của `mount` để giới hạn tìm kiếm trong các loại hệ thống tệp cụ thể. Một tệp mà người dùng hiện tại có thể đọc được sẽ có ít nhất là `4` ở chữ số đầu tiên của quyền và một tệp có thể thực thi được bởi nhóm sẽ có ít nhất là `1` ở chữ số thứ hai. Vì không quan tâm đến quyền của những người khác nên chúng ta có thể sử dụng `0` cho chữ số thứ ba. Sử dụng `-cmin N` để lọc các thay đổi thuộc tính gần đây (hãy nhớ rằng N được chỉ định theo phút). Vì thế nên câu trả lời sẽ là:

   ```
   find /mnt -fstype ext4 -perm -410 -cmin -120
   ```

4. Hãy tìm các tệp trống được tạo cách đây hơn 30 ngày và xuống dưới ít nhất hai cấp so với thư mục hiện tại.

   Tham số `-mindeep N` có thể được sử dụng để giới hạn tìm kiếm ở mức ít nhất là `N`, nhưng hãy nhớ rằng bạn phải tính cả thư mục hiện tại. Sử dụng `-empty` để kiểm tra các tệp trống và `-mtime N` để kiểm tra thời gian sửa đổi. Vì thế:

   ```
   find . -empty -mtime +30 -mindepth 3
   ```

5. Giả sử người dùng `carol` và `john` là một phần của nhóm `mkt`. Hãy tìm trong thư mục chính của

`john` bất kỳ tệp nào mà `` `carol `` cũng có thể đọc được.

Vì họ là thành viên của cùng một nhóm, chúng ta cần ít nhất một `r` (4) trên các quyền của nhóm và vì không quan tâm đến quyền của những người khác nên câu trả lời sẽ là:

```
find /home/john -perm -040
```

Ấn bản

PDF được tạo vào: 2023-10-16

Tài liệu Học tập LPI là một sáng kiến của Linux Professional Institute (https://lpi.org). Tài liệu Học tập và các Bản dịch của chúng có thể được tìm thấy tại https://learning.lpi.org.

Đối với các câu hỏi và nhận xét về ấn bản này cũng như về toàn bộ dự án, hãy gửi email tới: learning@lpi.org.