



Linux Essentials

Phiên bản 1.6
Tiếng Việt

010

Table of Contents

CHỦ ĐỀ 1: CỘNG ĐỒNG LINUX VÀ SỰ NGHIỆP TRONG MÃ NGUỒN MỞ	1
1.1 Sự phát triển của Linux và các hệ điều hành phổ biến	2
1.1 Bài 1	3
Giới thiệu	3
Các Bản Phân phối	4
Hệ thống nhúng	5
Linux và Đám mây	7
Bài tập Hướng dẫn	8
Bài tập Mở rộng	9
Tóm tắt	10
Đáp án Bài tập Hướng dẫn	11
Đáp án Bài tập Mở rộng	13
1.2 Các Ứng dụng Mã Nguồn Mở chính	14
1.2 Bài 1	15
Giới thiệu	15
Gói Phần mềm	15
Cài đặt Gói	16
Gỡ bỏ Gói	19
Ứng dụng Văn phòng	21
Trình duyệt Web	22
Truyền thông Đa phương tiện	22
Chương trình Máy chủ	23
Chia sẻ Dữ liệu	24
Quản Trị Mạng	25
Ngôn ngữ Lập trình	26
Bài tập Hướng dẫn	29
Bài tập Mở rộng	30
Tóm tắt	31
Đáp án Bài tập Hướng dẫn	32
Đáp án Bài tập Mở rộng	34
1.3 Phần mềm Mã Nguồn Mở và Giấy phép	35
1.3 Bài 1	36
Giới thiệu	36
Định nghĩa về Phần mềm Mã nguồn Mở và Tự do	36
Giấy phép	39
Mô hình kinh doanh trong Mã nguồn mở	43
Bài tập Hướng dẫn	46
Bài tập Mở rộng	47

Tóm tắt	48
Đáp án Bài tập Hướng dẫn	49
Đáp án Bài tập Mở rộng	50
1.4 Kỹ năng CNTT và Làm việc trên Linux	52
1.4 Bài 1	53
Giới thiệu	53
Giao diện Người dùng Linux	54
Công dụng của Linux trong Công nghiệp	56
Các vấn đề về Quyền riêng tư khi sử dụng Internet	57
Mã hóa	61
Bài tập Hướng dẫn	64
Bài tập Mở rộng	66
Tóm tắt	67
Đáp án Bài tập Hướng dẫn	68
Đáp án Bài tập Mở rộng	70
CHỦ ĐỀ 2: TÌM MỘT LỖI ĐI TRONG HỆ THỐNG LINUX	71
2.1 Khái niệm cơ bản về dòng lệnh	72
2.1 Bài 1	73
Giới thiệu	73
Cấu trúc của Dòng lệnh	75
Các loại Hành vi của Lệnh	76
Trích dẫn	76
Bài tập Hướng dẫn	80
Bài tập Mở rộng	82
Tóm tắt	83
Đáp án Bài tập Hướng dẫn	84
Đáp án Bài tập Mở rộng	85
2.1 Bài 2	86
Giới thiệu	86
Biến	86
Thao tác với Biến	87
Bài tập Hướng dẫn	92
Bài tập Mở rộng	93
Tóm tắt	94
Đáp án Bài tập Hướng dẫn	95
Đáp án Bài tập Mở rộng	96
2.2 Sử dụng Dòng lệnh để Nhận Trợ giúp	98
2.2 Bài 1	99
Giới thiệu	99
Nhận Trợ giúp trên Dòng lệnh	99

Định vị Tệp	102
Bài tập Hướng dẫn	104
Bài tập Mở rộng	106
Tóm tắt	107
Đáp án Bài tập Hướng dẫn	108
Đáp án Bài tập Mở rộng	111
2.3 Sử dụng Thư mục và Tệp liệt kê	113
2.3 Bài 1	114
Giới thiệu	114
Tệp và Thư mục	114
Tên Tệp và Thư mục	115
Điều hướng Hệ thống tệp	115
Đường dẫn Tuyệt đối và Tương đối	117
Bài tập Hướng dẫn	119
Bài tập Mở rộng	121
Tóm tắt	122
Đáp án Bài tập Hướng dẫn	123
Đáp án Bài tập Mở rộng	126
2.3 Bài 2	127
Giới thiệu	127
Thư mục Chính	127
Đường dẫn Tương đối Đặc biệt cho Trang chủ	129
Đường dẫn tệp từ Tương-đối-đến-Trang-chủ	130
Tệp và Thư mục ẩn	131
Tùy chọn Danh sách Dài	132
Các Tùy chọn ls Bổ sung	132
Đệ quy trong Bash	133
Bài tập Hướng dẫn	135
Bài tập Mở rộng	137
Tóm tắt	138
Đáp án Bài tập Hướng dẫn	139
Đáp án Bài tập Mở rộng	141
2.4 Tạo, di chuyển và xóa Tệp	142
2.4 Bài 1	143
Giới thiệu	143
Phân biệt Chữ hoa và Chữ thường	144
Tạo Thư mục	144
Tạo Tệp	146
Đổi tên Tệp	147
Di chuyển Tệp	148

Xóa Tập và Thư mục	149
Sao chép Tập và Thư mục	151
Khớp mẫu khối	153
Bài tập Hướng dẫn	158
Bài tập Mở rộng	160
Tóm tắt	161
Đáp án Bài tập Hướng dẫn	163
Đáp án Bài tập Mở rộng	166
CHỦ ĐỀ 3: SỨC MẠNH CỦA DÒNG LỆNH	168
3.1 Lưu trữ Tập trên Dòng lệnh	169
3.1 Bài 1	170
Giới thiệu	170
Công cụ Nén	171
Công cụ Lưu trữ	173
Quản lý tập ZIP	177
Bài tập Hướng dẫn	179
Bài tập Mở rộng	180
Tóm tắt	181
Đáp án Bài tập Hướng dẫn	183
Đáp án Bài tập Mở rộng	185
3.2 Tìm kiếm và trích xuất dữ liệu từ Tập	186
3.2 Bài 1	187
Giới thiệu	187
Chuyển hướng I/O	187
Đường ống Dòng lệnh	192
Bài tập Hướng dẫn	194
Bài tập Mở rộng	195
Tóm tắt	196
Đáp án Bài tập Hướng dẫn	197
Đáp án Bài tập Mở rộng	199
3.2 Bài 2	200
Giới thiệu	200
Tìm kiếm trong Tập bằng <code>grep</code>	200
Biểu thức Chính quy	201
Bài tập Hướng dẫn	205
Bài tập Mở rộng	206
Tóm tắt	207
Đáp án Bài tập Hướng dẫn	208
Đáp án Bài tập Mở rộng	210
3.3 Biến Lệnh thành Tập lệnh	212

3.3 Bài 1	213
Giới thiệu	213
Đầu ra In	213
Tạo Tệp lệnh Thực thi	214
Lệnh và PATH	214
Thực thi các Quyền	215
Định nghĩa Trình thông dịch	216
Biến	217
Sử dụng Trích dẫn với các Biến	219
Đối số	220
Trả về Số của Đối số	221
Logic có điều kiện	222
Bài tập Hướng dẫn	224
Bài tập Mở rộng	226
Tóm tắt	227
Đáp án Bài tập Hướng dẫn	228
Đáp án Bài tập Mở rộng	230
3.3 Bài 2	232
Giới thiệu	232
Mã Thoát	233
Xử lý nhiều Đối số	235
Vòng lặp For	236
Sử dụng Biểu thức Chính quy để Thực hiện Kiểm tra Lỗi	238
Bài tập Hướng dẫn	241
Bài tập Mở rộng	243
Tóm tắt	244
Đáp án Bài tập Hướng dẫn	245
Đáp án Bài tập Mở rộng	247
CHỦ ĐỀ 4: HỆ ĐIỀU HÀNH LINUX	248
4.1 Chọn Hệ điều hành	249
4.1 Bài 1	250
Giới thiệu	250
Hệ Điều hành là gì	250
Chọn một Bản Phân phối Linux	251
Hệ điều hành không phải Linux	255
Bài tập Hướng dẫn	258
Bài tập Mở rộng	260
Tóm tắt	261
Đáp án Bài tập Hướng dẫn	262
Đáp án Bài tập Mở rộng	264

4.2 Hiểu phần cứng của Máy tính	265
4.2 Bài 1	266
Giới thiệu	266
Nguồn Điện	267
Bo mạch chủ	267
Bộ nhớ	268
Bộ Vi Xử lý	269
Lưu trữ	272
Phân vùng	273
Thiết bị ngoại vi	274
Trình Điều khiển và Tập Thiết bị	274
Bài tập Hướng dẫn	277
Bài tập Mở rộng	278
Tóm tắt	279
Đáp án Bài tập Hướng dẫn	280
Đáp án Bài tập Mở rộng	282
4.3 Nơi lưu trữ dữ liệu	283
4.3 Bài 1	284
Giới thiệu	284
Các Chương trình và Cấu hình của chúng	285
Nhân Linux	289
Thiết bị Phần Cứng	292
Bộ nhớ và các loại Bộ nhớ	293
Bài tập Hướng dẫn	296
Bài tập Mở rộng	297
Tóm tắt	298
Đáp án Bài tập Hướng dẫn	300
Đáp án Bài tập Mở rộng	302
4.3 Bài 2	303
Giới thiệu	303
Quy trình	303
Ghi nhật ký Hệ thống và Thông báo của Hệ thống	307
Bài tập Hướng dẫn	313
Bài tập Mở rộng	316
Tóm tắt	318
Đáp án Bài tập Hướng dẫn	320
Đáp án Bài tập Mở rộng	323
4.4 Máy tính của bạn trên mạng	325
4.4 Bài 1	326
Giới thiệu	326

Mạng Tầng Liên kết	327
Mạng IPv4	328
Mạng IPv6	333
DNS	336
Ổ nổi	338
Bài tập Hướng dẫn	340
Bài tập Mở rộng	341
Tóm tắt	342
Đáp án Bài tập Hướng dẫn	343
Đáp án Bài tập Mở rộng	344
CHỦ ĐỀ 5: QUYỀN BẢO MẬT VÀ TỆP	346
5.1 Bảo mật Cơ bản và Xác định Loại Người dùng	347
5.1 Bài 1	348
Giới thiệu	348
Tài khoản	349
Nhận Thông tin về Người dùng của bạn	352
Chuyển đổi Người dùng và Nâng cao Đặc quyền	354
Tệp Kiểm soát Truy cập	355
Bài tập Hướng dẫn	363
Bài tập Mở rộng	365
Tóm tắt	366
Đáp án Bài tập Hướng dẫn	368
Đáp án Bài tập Mở rộng	370
5.2 Tạo người dùng và nhóm	372
5.2 Bài 1	373
Giới thiệu	373
Tệp /etc/passwd	374
Tệp /etc/group	375
Tệp /etc/shadow	375
Tệp /etc/gshadow	376
Thêm và Xóa Tài khoản Người dùng	377
Thư mục Skeleton	379
Thêm và Xóa Nhóm	380
Lệnh passwd	380
Bài tập Hướng dẫn	382
Bài tập Mở rộng	384
Tóm tắt	385
Đáp án Bài tập Hướng dẫn	386
Đáp án Bài tập Mở rộng	388
5.3 Quản lý Quyền và Quyền sở hữu Tệp	391

5.3 Bài 1	392
Giới thiệu	392
Truy vấn Thông tin về Tập và Thư mục	392
Còn Thư mục thì sao?	394
Xem Tập ẩn	394
Tìm hiểu các loại Tập	395
Hiểu về Quyền	396
Sửa đổi Quyền trong Tập	398
Chế độ Tượng trưng	399
Chế độ Số	400
Sửa đổi Quyền Sở hữu Tập	401
Nhóm Truy vấn	402
Quyền Đặc biệt	403
Bài tập Hướng dẫn	406
Bài tập Mở rộng	408
Tóm tắt	409
Đáp án Bài tập Hướng dẫn	410
Đáp án Bài tập Mở rộng	413
5.4 Các Thư mục và Tập đặc biệt	416
5.4 Bài 1	417
Giới thiệu	417
Tập Tạm thời	417
Hiểu về Liên kết	419
Bài tập Hướng dẫn	424
Bài tập Mở rộng	425
Tóm tắt	428
Đáp án Bài tập Hướng dẫn	429
Đáp án Bài tập Mở rộng	430
Ấn bản	434



**Linux
Professional
Institute**

Chủ đề 1: Cộng đồng Linux và Sự nghiệp trong Mã Nguồn Mở



**Linux
Professional
Institute**

1.1 Sự phát triển của Linux và các hệ điều hành phổ biến

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 1.1](#)

Khối lượng

2

Các lĩnh vực kiến thức chính

- Các bản Phân phối
- Hệ thống nhúng
- Linux trong Đám mây

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

- Debian, Ubuntu (LTS)
- CentOS, openSUSE, Red Hat, SUSE
- Linux Mint, Scientific Linux
- Raspberry Pi, Raspbian
- Android



Linux
Professional
Institute

1.1 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	1 Cộng đồng Linux và Sự nghiệp trong lĩnh vực Mã nguồn mở
Mục tiêu:	1.1 Sự phát triển của Linux và các hệ điều hành phổ biến
Bài:	1 trên 1

Giới thiệu

Linux là một trong những hệ điều hành phổ biến nhất hiện nay trên thế giới. Sự phát triển của nó được bắt đầu từ năm 1991 bởi Linus Torvalds. Linux được lấy cảm hứng từ Unix - một hệ điều hành khác được phát triển vào những năm 1970 bởi Phòng thí nghiệm AT&T. Unix vốn được hướng tới các loại máy tính nhỏ. Vào thời điểm đó, máy tính được coi là "nhỏ" khi người ta không cần đến cả một hội trường có gắn điều hòa để chứa nó và có giá dưới một triệu đô la. Sau này thì máy tính "nhỏ" tức là khi nó có thể được nâng lên bởi hai người. Vào thời điểm đó, một hệ thống Unix giá cả phải chăng không phải lúc nào cũng có sẵn trên các máy tính (ví dụ như máy tính văn phòng - vốn có xu hướng vận hành dựa trên nền tảng x86). Do đó, Linus, khi đó còn là một sinh viên, đã bắt đầu phát triển một hệ điều hành giống Unix mà đáng ra sẽ chạy trên nền tảng x86.

Linux phần lớn sử dụng cùng các nguyên tắc và ý tưởng cơ bản của Unix, nhưng bản thân Linux không có chứa mã Unix vì nó là một dự án độc lập. Linux không được hỗ trợ bởi một công ty riêng lẻ nào mà bởi một cộng đồng lập trình viên quốc tế. Bất kỳ ai cũng có thể sử dụng được Linux mà không bị hạn chế bởi nó được cung cấp hoàn toàn miễn phí.

Các Bản Phân phối

Một *bản phân phối* Linux là một gói bao gồm *nhân* (kernel) Linux và một loạt các ứng dụng được tuyển chọn và duy trì bởi một công ty hoặc một cộng đồng người dùng. Mục tiêu của một bản phân phối là tối ưu hóa nhân và các ứng dụng chạy trên hệ điều hành cho một trường hợp sử dụng hoặc một nhóm người dùng nhất định. Các bản phân phối sẽ thường bao gồm nhiều các công cụ dành riêng cho từng bản phân phối để cài đặt phần mềm và quản trị hệ thống. Đây là lý do tại sao một số bản phân phối chủ yếu chỉ được sử dụng cho môi trường máy tính để bàn nơi chúng cần phải dễ sử dụng, trong khi những bản phân phối khác lại chủ yếu được dùng để chạy trên các máy chủ nhằm mục đích khai thác các tài nguyên có sẵn một cách hiệu quả nhất có thể.

Một cách khác để phân loại các bản phân phối là đối chiếu về *Dòng phân phối* của chúng. Các bản phân phối thuộc Dòng phân phối Debian sử dụng trình quản lý gói `dpkg` để quản lý phần mềm chạy trên hệ điều hành. Các gói có thể cài đặt cùng với trình quản lý gói được duy trì bởi các tình nguyện viên của cộng đồng phân phối. Các tình nguyện viên này sử dụng định dạng gói `deb` để chỉ định cách phần mềm được cài đặt trên hệ điều hành và cấu hình mặc định của phần mềm. Giống như một bản phân phối, gói là một gói phần mềm với cấu hình cũng như tài liệu tương ứng giúp người dùng cài đặt, cập nhật và sử dụng phần mềm dễ dàng hơn.

Bản phân phối *Debian GNU/Linux* là bản phân phối lớn nhất của Dòng phân phối Debian. Dự án Debian GNU/Linux được khởi xướng vào năm 1993 bởi Ian Murdock. Ngày nay, hàng nghìn tình nguyện viên đang làm việc cho dự án này. Debian GNU/Linux hướng tới mục đích cung cấp một hệ điều hành đáng tin cậy tuyệt đối. Nó cũng đóng vai trò quảng bá tầm nhìn của Richard Stallman về một hệ điều hành tôn trọng quyền tự do của người dùng trong việc chạy, nghiên cứu, phân phối và cải thiện phần mềm. Đây là lý do tại sao nó không mặc định cung cấp bất kỳ phần mềm độc quyền nào.

Ubuntu là một bản phân phối dựa trên Debian khác đáng để nhắc tới. Ubuntu được tạo ra bởi Mark Shuttleworth và đội ngũ của ông vào năm 2004 với sứ mệnh mang đến một môi trường máy tính để bàn Linux dễ sử dụng. Nhiệm vụ của Ubuntu là cung cấp phần mềm tự do cho mọi người trên toàn thế giới, cũng như cắt giảm chi phí cho các dịch vụ chuyên nghiệp. Bản phân phối này có lịch phát hành sáu tháng một lần với bản phát hành hỗ trợ dài hạn mỗi hai năm một lần.

Red Hat là một bản phân phối Linux được phát triển và duy trì bởi công ty phần mềm có cùng tên và được IBM mua lại vào năm 2019. Bản phân phối Red Hat Linux được bắt đầu vào năm 1994 và được đổi tên thành *Red Hat Enterprise Linux* vào năm 2003, thường được viết tắt là RHEL. Nó được cung cấp cho các công ty như một giải pháp doanh nghiệp đáng tin cậy được Red Hat hỗ trợ và được đính kèm các phần mềm nhằm mục đích giúp cho việc sử dụng Linux trở nên dễ dàng hơn trong môi trường máy chủ chuyên nghiệp. Một số thành phần của nó có yêu cầu đăng ký theo dõi tính phí hoặc giấy phép. Dự án *CentOS* sử dụng mã nguồn có sẵn miễn phí của Red Hat Enterprise Linux và biên soạn nó thành một bản phân phối tuy hoàn toàn miễn phí nhưng lại không đi kèm

hỗ trợ thương mại.

Cả RHEL và CentOS đều được tối ưu hóa để sử dụng trong môi trường máy chủ. Dự án *Fedora* được thành lập vào năm 2003 và tạo ra một bản phân phối Linux nhắm tới dòng máy tính để bàn. Red Hat khởi xướng và duy trì việc phân phối Fedora kể từ đó. Fedora rất tiên tiến và nhanh chóng áp dụng các công nghệ mới, đôi khi còn được coi là nơi để thử nghiệm các công nghệ mới mà sau này có thể được đưa vào RHEL. Tất cả các bản phân phối dựa trên Red Hat đều sử dụng định dạng gói rpm. Công ty SUSE được thành lập năm 1992 tại Đức với tư cách là nhà cung cấp dịch vụ Unix. Phiên bản đầu tiên của *SUSE Linux* được phát hành vào năm 1994. Trong những năm qua, SUSE Linux hầu hết được biết đến với công cụ cấu hình YaST. Công cụ này cho phép quản trị viên cài đặt và cấu hình phần mềm cũng như phần cứng, thiết lập máy chủ và mạng. Tương tự như RHEL, SUSE phát hành *SUSE Linux Enterprise Server*, tức phiên bản thương mại của họ. Phiên bản này ít được phát hành hơn và phù hợp cho việc triển khai doanh nghiệp và sản xuất. Nó được phân phối dưới dạng một máy chủ cũng như một môi trường máy tính để bàn, với các gói dành cho mọi mục đích sử dụng. Năm 2004, SUSE phát hành dự án *openSUSE*, mở ra các nhà phát triển và người dùng nhiều cơ hội thử nghiệm và phát triển hệ thống hơn nữa. Bản phân phối openSUSE có sẵn để tải về miễn phí.

Nhiều bản phân phối độc lập đã được phát hành trong suốt nhiều năm qua, một trong số đó dựa trên Red Hat hoặc Ubuntu, một số thì được thiết kế để cải thiện một quyền sở hữu cụ thể của hệ thống hoặc phần cứng. Có những bản phân phối được xây dựng với các chức năng cụ thể như *QubesOS* - một môi trường máy tính để bàn có an ninh cao hoặc như *Kali Linux* - cung cấp một môi trường để khai thác các lỗ hổng phần mềm, chủ yếu được sử dụng bởi những các Tester xâm nhập. Gần đây, nhiều bản phân phối Linux siêu nhỏ khác nhau được thiết kế để chạy riêng trong các vật chứa Linux (Linux container), chẳng hạn như Docker. Ngoài ra còn có các bản phân phối được xây dựng riêng cho các thành phần của hệ thống nhúng và thậm chí cả các thiết bị thông minh.

Hệ thống nhúng

Hệ thống nhúng là sự kết hợp giữa phần cứng và phần mềm máy tính được thiết kế để có một chức năng cụ thể trong một hệ thống lớn hơn. Thông thường chúng là một phần của các thiết bị khác và giúp điều khiển các thiết bị này. Các hệ thống nhúng được tìm thấy trong các ứng dụng ô tô, y tế và thậm chí cả quân sự. Do số lượng các ứng dụng rất nhiều và đa dạng, nhiều hệ điều hành dựa trên nhân Linux đã được phát triển để sử dụng trong các hệ thống nhúng. Một phần quan trọng của các thiết bị thông minh có một hệ điều hành dựa trên nhân Linux chạy trên đó.

Do đó, hệ thống nhúng phải đi kèm với phần mềm nhúng. Mục đích của các phần mềm này là truy cập vào phần cứng và khiến nó trở nên khả dụng. Những ưu điểm lớn của Linux so với bất kỳ phần mềm nhúng độc quyền nào khác là khả năng tương thích, sự phát triển, tính năng hỗ trợ lẫn nhau trên nền tảng của rất nhiều các nhà cung cấp cũng như việc không tính phí cấp phép. Hai trong số các dự án phần mềm nhúng phổ biến nhất là Android - được sử dụng chủ yếu trên điện

thoại di động của nhiều nhà cung cấp và Raspbian - được sử dụng chủ yếu trên Raspberry Pi.

Android

Android chủ yếu được coi là một hệ điều hành di động được phát triển bởi Google. Android Inc. được thành lập vào năm 2003 tại Palo Alto, California. Ban đầu, công ty này đã tạo ra một hệ điều hành vốn để chạy trên máy ảnh kỹ thuật số. Vào năm 2005, Google mua lại Android Inc. và đã phát triển nó trở thành một trong những hệ điều hành di động lớn nhất mọi thời đại. Cơ sở của Android là phiên bản đã được sửa đổi của nhân Linux với các phần mềm mã nguồn mở bổ sung. Hệ điều hành này chủ yếu được phát triển cho các thiết bị màn hình cảm ứng, nhưng Google đã phát triển cả các phiên bản dành cho TV và đồng hồ đeo tay. Các phiên bản khác nhau của Android cũng được phát triển cho máy chơi game, máy ảnh kỹ thuật số cũng như máy tính cá nhân. Android được cung cấp miễn phí trong mã nguồn mở dưới dạng *Dự án Mã nguồn mở Android* (Android Open Source Project - AOSP). Google cung cấp một loạt các thành phần độc quyền bên cạnh lõi mã nguồn mở của Android. Các thành phần này bao gồm các ứng dụng như Google Calendar, Google Maps, Google Mail, trình duyệt Chrome cũng như Cửa hàng Google Play với mục đích hỗ trợ việc cài đặt ứng dụng trở nên dễ dàng hơn. Hầu hết người dùng coi những công cụ này là một phần không thể thiếu trong trải nghiệm Android của họ. Do đó, hầu hết tất cả các thiết bị di động chạy Android ở Châu Âu và Châu Mỹ đều có phần mềm độc quyền của Google.

Android trên thiết bị nhúng có khá nhiều lợi thế. Hệ điều hành này có tính trực quan, có giao diện người dùng mang tính hình ảnh cao rất dễ sử dụng. Nó được hỗ trợ bởi một cộng đồng các nhà phát triển rất rộng nên người dùng có thể dễ dàng tìm kiếm sự trợ giúp trong việc phát triển. Thêm vào đó, Android cũng được hỗ trợ bởi đa số các nhà cung cấp phần cứng thông qua trình điều khiển Android; do đó, việc tạo mẫu một hệ thống hoàn chỉnh là rất dễ dàng và tiết kiệm chi phí.

Raspbian và Raspberry Pi

Raspberry Pi là một máy tính có kích thước chỉ bằng một chiếc thẻ tín dụng, giá thành thấp và có thể hoạt động như một chiếc máy tính để bàn đầy đủ chức năng, nhưng cũng có thể được sử dụng trong một hệ thống Linux nhúng. Raspberry Pi được phát triển bởi Raspberry Pi Foundation, vốn là một tổ chức từ thiện giáo dục có trụ sở tại Vương quốc Anh. Mục đích chủ yếu của Raspberry Pi là để dạy những người trẻ tuổi học lập trình và hiểu chức năng của máy tính. Raspberry Pi có thể được thiết kế và lập trình để thực hiện các tác vụ hoặc hoạt động được yêu cầu vốn là một phần của một hệ thống phức tạp hơn rất nhiều.

Điểm đặc biệt của Raspberry Pi là việc nó bao gồm một bộ chân tín hiệu kỹ thuật số Đầu vào và Đầu ra (GPIO - General Purpose Input-Output) có thể được sử dụng để gắn các thiết bị điện tử và bảng mở rộng. Điều này cho phép Raspberry Pi có thể được sử dụng để làm nền tảng phát triển phần cứng. Mặc dù được thiết kế cho mục đích giáo dục, nhưng ngày nay Raspberry Pi được sử

dụng trong các dự án DIY - Tự tay làm lấy - khác nhau, cũng như để tạo mẫu công nghiệp khi phát triển các hệ thống nhúng.

Raspberry Pi sử dụng bộ vi xử lý ARM. Nhiều hệ điều hành khác nhau, bao gồm cả Linux, đều chạy trên Raspberry Pi. Vì Raspberry Pi không chứa ổ cứng nên hệ điều hành được khởi động từ thẻ nhớ SD. Một trong những bản phân phối Linux nổi bật nhất dành cho Raspberry Pi là *Raspbian*. Như cái tên đã cho hay, Raspbian cũng thuộc Dòng phân phối Debian. Nó được tùy chỉnh để cài đặt trên phần cứng Raspberry Pi và cung cấp hơn 35.000 gói tối ưu hóa cho môi trường này. Ngoài Raspbian còn có nhiều bản phân phối Linux khác dành cho Raspberry Pi, chẳng hạn như Kodi - bản phân phối đã biến Raspberry Pi thành một trung tâm truyền thông.

Linux và Đám mây

Thuật ngữ *điện toán đám mây* đề cập đến một cách tiêu thụ tài nguyên máy tính được tiêu chuẩn hóa thông qua việc mua chúng từ nhà cung cấp đám mây công cộng hoặc bằng cách chạy một đám mây riêng. Theo báo cáo vào năm 2017, Linux chạy 90% khối lượng công việc trên đám mây công cộng. Mọi nhà cung cấp đám mây, từ *Amazon Web Services (AWS)* đến *Google Cloud Platform (GCP)*, đều cung cấp các dạng Linux khác nhau. Ngay cả Microsoft cũng cung cấp các máy ảo dựa trên Linux trong đám mây *Azure* của họ ngày nay.

Thông thường, Linux được cung cấp như một phần của dịch vụ *Cơ sở hạ tầng dưới dạng Dịch vụ (IaaS - Infrastructure as a Service)*. Các trường hợp IaaS là máy ảo được cung cấp trong vòng vài phút trên đám mây. Khi khởi động một trường hợp IaaS, một hình ảnh sẽ được chọn để chứa dữ liệu được sử dụng để triển khai sang trường hợp mới. Các nhà cung cấp dịch vụ đám mây cung cấp nhiều các hình ảnh khác nhau chứa các bản cài đặt sẵn của cả các bản phân phối Linux phổ biến lẫn các phiên bản Linux của riêng người dùng. Người dùng đám mây có thể chọn một hình ảnh chứa bản phân phối ưa thích của họ và truy cập một trường hợp đám mây chứa bản phân phối này ngay sau đó. Hầu hết các nhà cung cấp đám mây đều thêm các công cụ vào hình ảnh của họ để điều chỉnh việc cài đặt cho mỗi trường hợp đám mây nhất định. Ví dụ, những công cụ này có thể mở rộng hệ thống tệp của hình ảnh để phù hợp với ổ cứng thực của máy ảo.

Bài tập Hướng dẫn

1. Debian GNU/Linux khác với Ubuntu ở điểm nào? Hãy chỉ ra hai khía cạnh khác biệt.

2. Linux được sử dụng phổ biến nhất ở môi trường/nền tảng nào? Chỉ ra ba môi trường/nền tảng khác nhau và nêu tên một bản phân phối có thể sử dụng được ở từng môi trường/ nền tảng đó.

3. Bạn đang dự định cài đặt một bản phân phối Linux trong một môi trường mới. Kể tên bốn việc mà bạn nên xem xét khi chọn một bản phân phối.

4. Kể tên ba thiết bị chạy hệ điều hành Android (không tính điện thoại thông minh).

5. Giải thích ba ưu điểm chính của điện toán đám mây.

Bài tập Mở rộng

1. Xem xét khía cạnh chi phí và hiệu suất, bản phân phối nào phù hợp nhất cho một doanh nghiệp có mong muốn giảm thiểu chi phí cấp phép, trong khi vẫn giữ hiệu suất ở mức cao nhất? Hãy đưa ra và giải thích lý do.

2. Những ưu điểm chính của Raspberry Pi là gì và chúng có thể đảm nhận những chức năng nào trong kinh doanh?

3. Amazon Cloud Services và Google Cloud cung cấp những bản phân phối nào? Kể tên ít nhất ba bản phân phối phổ biến và hai bản phân phối khác ngoài chúng.

Tóm tắt

Trong bài học này, bạn đã học về:

- Những bản phân phối mà Linux có
- Hệ thống nhúng Linux là gì
- Các hệ thống nhúng Linux được sử dụng như thế nào
- Các khả năng ứng dụng khác nhau của Android
- Các ứng dụng khác nhau của Raspberry Pi
- Điện toán Đám mây là gì
- Linux đóng vai trò gì trong điện toán đám mây

Đáp án Bài tập Hướng dẫn

1. Debian GNU/Linux khác với Ubuntu như thế nào? Hãy chỉ ra hai khía cạnh khác biệt.

Ubuntu được dựa trên một bản snapshot của Debian, do đó giữa hai bên có nhiều điểm tương đồng. Tuy vậy, những khác biệt giữa chúng cũng khá đáng kể. Điểm đầu tiên sẽ là khả năng áp dụng cho người mới bắt đầu. Ubuntu được khuyên dùng cho người mới bắt đầu vì tính dễ sử dụng của nó, còn Debian lại được khuyên dùng cho những người dùng cao cấp hơn. Sự khác biệt chính nằm ở cấu hình người dùng phức tạp mà Ubuntu không yêu cầu trong quá trình cài đặt.

Một điểm khác biệt nữa là tính ổn định của mỗi bản phân phối. Debian được coi là ổn định hơn so với Ubuntu do Debian nhận được ít bản cập nhật kiểm tra chi tiết hơn và toàn bộ hệ điều hành của nó ổn định hơn. Mặt khác, Ubuntu cho phép người dùng sử dụng các phiên bản phần mềm mới nhất và tất cả các công nghệ mới.

2. Linux được sử dụng phổ biến nhất ở môi trường/nền tảng nào? Chỉ ra ba môi trường/nền tảng khác nhau và nêu tên một bản phân phối có thể sử dụng được ở từng môi trường/ nền tảng đó.

Một số môi trường/nền tảng phổ biến nhất là điện thoại thông minh, máy tính để bàn và máy chủ. Trên điện thoại thông minh, Linux có thể được sử dụng bởi các bản phân phối như Android. Trên máy tính để bàn và máy chủ, nó có thể được sử dụng bởi bất kỳ bản phân phối nào phù hợp nhất với chức năng của máy đó, từ Debian, Ubuntu đến CentOS và Red Hat Enterprise Linux.

3. Bạn đang dự định cài đặt một bản phân phối Linux trong một môi trường mới. Kể tên bốn việc mà bạn nên xem xét khi chọn một bản phân phối.

Khi chọn một bản phân phối, các yếu tố chính cần được xem xét là chi phí, hiệu suất, khả năng mở rộng, mức độ ổn định và nhu cầu phần cứng của hệ thống.

4. Kể tên ba thiết bị chạy hệ điều hành Android (không tính điện thoại thông minh).

Một số thiết bị khác chạy Android là TV thông minh, máy tính bảng, Android Auto và đồng hồ thông minh.

5. Giải thích ba ưu điểm chính của điện toán đám mây.

Ưu điểm chính của điện toán đám mây là tính linh hoạt, dễ khôi phục và chi phí sử dụng thấp. Các dịch vụ dựa trên đám mây rất dễ triển khai và mở rộng quy mô, tùy thuộc vào yêu cầu kinh doanh. Điện toán đám mây có một lợi thế lớn trong các giải pháp sao lưu và phục hồi vì nó cho phép các doanh nghiệp phục hồi nhanh hơn và để lại ít hậu quả hơn sau sự cố. Hơn nữa, nó còn

làm giảm chi phí vận hành vì nó cho phép người dùng chỉ phải trả tiền cho các tài nguyên mà doanh nghiệp sử dụng thông qua mô hình thuê bao đăng ký theo dõi.

Đáp án Bài tập Mở rộng

1. Xem xét khía cạnh chi phí và hiệu suất, bản phân phối nào phù hợp nhất cho một doanh nghiệp có mong muốn giảm thiểu chi phí cấp phép, trong khi vẫn giữ hiệu suất ở mức cao nhất? Hãy đưa ra và giải thích lý do.

Một trong những bản phân phối phù hợp nhất cho trường hợp này là CentOS. Lý do là bởi nó kết hợp tất cả các sản phẩm của Red Hat; các sản phẩm này sau đó được sử dụng sâu hơn trong hệ điều hành thương mại của họ, trong khi vẫn không phải trả phí. Tương tự, các bản phát hành Hỗ trợ dài hạn (LTS) của Ubuntu đảm bảo sự hỗ trợ của họ trong một khoảng thời gian dài hơn. Các phiên bản ổn định của Debian GNU/Linux cũng thường được sử dụng trong môi trường doanh nghiệp.

2. Những ưu điểm chính của Raspberry Pi là gì và chúng có thể đảm nhận những chức năng nào trong kinh doanh?

Raspberry Pi có kích thước nhỏ hơn trong khi vẫn có khả năng hoạt động như một máy tính thông thường. Hơn nữa, nó có giá thành rẻ và có thể xử lý lưu lượng truy cập web và nhiều chức năng khác. Nó có thể được sử dụng làm máy chủ, tường lửa và bo mạch chính cho robot và nhiều thiết bị nhỏ khác.

3. Amazon Cloud Services và Google Cloud cung cấp những bản phân phối nào? Kể tên ít nhất ba bản phân phối phổ biến và hai bản phân phối khác ngoài chúng.

Các bản phân phối phổ biến giữa Amazon và Google Cloud Services là Ubuntu, CentOS và Red Hat Enterprise Linux. Mỗi nhà cung cấp đám mây cũng cung cấp các bản phân phối đặc biệt mà nhà cung cấp khác không có. Amazon có Amazon Linux và Kali Linux, trong khi Google cung cấp việc sử dụng FreeBSD và Windows Servers (Máy chủ Windows).



**Linux
Professional
Institute**

1.2 Các Ứng dụng Mã Nguồn Mở chính

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 1.2](#)

Khối lượng

2

Các lĩnh vực kiến thức chính

- Ứng dụng máy tính để bàn
- Ứng dụng Máy chủ
- Ngôn ngữ phát triển
- Các công cụ và kho lưu trữ quản lý gói

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

- OpenOffice.org, LibreOffice, Thunderbird, Firefox, GIMP
- Nextcloud, ownCloud
- Apache HTTPD, NGINX, MariaDB, MySQL, NFS, Samba
- C, Java, JavaScript, Perl, shell, Python, PHP
- dpkg, apt-get, rpm, yum



1.2 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	1 Cộng đồng Linux và Sự nghiệp trong lĩnh vực Mã nguồn mở
Mục tiêu:	1.2 Các ứng dụng chính của Mã nguồn mở
Bài:	1 trên 1

Giới thiệu

Ứng dụng là một chương trình máy tính có mục đích không trực tiếp liên quan tới hoạt động bên trong của máy tính mà với các tác vụ do người dùng thực hiện. Các bản phân phối của Linux cung cấp nhiều tùy chọn ứng dụng để thực hiện nhiều tác vụ khác nhau, chẳng hạn như ứng dụng văn phòng, trình duyệt web, trình phát đa phương tiện và trình chỉnh sửa, v.v. Thường sẽ có nhiều hơn một ứng dụng hoặc công cụ để thực hiện một công việc cụ thể, tùy thuộc vào việc người dùng sẽ chọn ứng dụng nào để phù hợp nhất với nhu cầu của họ.

Gói Phần mềm

Hầu như mọi bản phân phối của Linux đều có cung cấp một bộ ứng dụng mặc định đã được cài đặt sẵn. Bên cạnh đó, mỗi bản phân phối sẽ có một kho lưu trữ gói cùng với một bộ sưu tập lớn các ứng dụng có sẵn để cài đặt thông qua *trình quản lý gói* của nó. Mặc dù các bản phân phối khác nhau sẽ cung cấp các loại ứng dụng gần như giống nhau, vẫn có một số hệ thống quản lý gói khác nhau dành cho các bản phân phối tồn tại. Ví dụ: Debian, Ubuntu và Linux Mint sử dụng các công cụ `dpkg`, `apt-get` và `apt` để cài đặt các gói phần mềm, và các công cụ này thường được gọi chung là *Gói DEB*. Các bản phân phối như Red Hat, Fedora và CentOS thay vào đó lại sử dụng các lệnh

`rpm`, `yum` và `dnf` để cài đặt các *Gói RPM*. Vì mỗi Dòng phân phối có một cách đóng gói ứng dụng khác nhau, việc cài các gói từ kho lưu trữ chính xác được thiết kế cho từng bản phân phối là rất quan trọng. Người dùng cuối thường không phải lo lắng về những chi tiết này vì trình quản lý gói của bản phân phối sẽ tự chọn các gói phù hợp, các phần phụ thuộc bắt buộc và các bản cập nhật trong tương lai. Các Phần phụ thuộc là các gói phụ trợ cần thiết cho các chương trình. Ví dụ: nếu một thư viện cung cấp các chức năng để xử lý ảnh JPEG được nhiều chương trình sử dụng thì khả năng cao thư viện này sẽ có gói riêng của nó, và tất cả các ứng dụng sử dụng thư viện đó đều sẽ phụ thuộc vào gói này.

Các lệnh `dpkg` và `rpm` hoạt động trên các tệp riêng lẻ trong gói. Trên thực tế, hầu hết tất cả các tác vụ quản lý gói trên hệ thống sử dụng gói DEB đều được thực hiện bằng lệnh `apt-get` hoặc `apt`, hoặc trên hệ thống sử dụng gói RPM là bằng lệnh `yum` hoặc `dnf`. Các lệnh này sẽ làm việc với danh mục của các gói, có thể tải xuống các gói mới và phần phụ thuộc của chúng, đồng thời kiểm tra các phiên bản mới hơn của các gói đã được cài đặt.

Cài đặt Gói

Giả sử bạn đã nghe nói về một lệnh gọi là `figlet` dùng để in văn bản chữ lớn trên cửa sổ dòng lệnh (chương trình giao diện cửa sổ dòng lệnh) và bạn muốn thử sử dụng nó. Tuy nhiên, bạn nhận được thông báo sau sau khi thực hiện lệnh `figlet`:

```
$ figlet
-bash: figlet: command not found
```

Tức là có lẽ gói này chưa được cài đặt trên hệ thống của bạn. Nếu bản phân phối của bạn làm việc với các gói DEB, bạn có thể tìm kiếm các kho lưu trữ của nó bằng cách sử dụng `apt-cache search package_name` hoặc `apt search package_name`. Lệnh `apt-cache` được dùng để tìm kiếm các gói và liệt kê thông tin về các gói có sẵn. Lệnh sau đây dùng để tìm kiếm sự xuất hiện bất kỳ của thuật ngữ “`figlet`” nào trong tên và mô tả của gói:

```
$ apt-cache search figlet
figlet - Make large character ASCII banners out of ordinary text
```

Quá trình tìm kiếm đã xác định một gói có tên *figlet* tương ứng với lệnh bị thiếu. Việc cài đặt và gỡ bỏ một gói yêu cầu các quyền đặc biệt chỉ được cấp cho quản trị viên hệ thống: tức người dùng có tên `root` (gốc). Trên các hệ thống máy tính để bàn, người dùng phổ thông có thể cài đặt hoặc gỡ bỏ các gói bằng cách thêm lệnh `sudo` vào các lệnh cài đặt/gỡ bỏ. Tác vụ này sẽ yêu cầu mật khẩu để có thể tiếp tục. Đối với các gói DEB, quá trình cài đặt sẽ được thực hiện bằng lệnh `apt-get install package_name` hoặc `apt install package_name`:

```
$ sudo apt-get install figlet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  figlet
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
```

Đến đây, gói sẽ được tải xuống và cài đặt trên hệ thống. Bất kỳ phần phụ thuộc nào mà gói cần cũng sẽ được tải xuống và cài đặt:

```
Need to get 184 kB of archives.
After this operation, 741 kB of additional disk space will be used.
Get:1 http://archive.raspbian.org/raspbian stretch/main armhf figlet armhf 2.2.5-2 [184 kB]
Fetched 184 kB in 0s (213 kB/s)
Selecting previously unselected package figlet.
(Reading database ... 115701 files and directories currently installed.)
Preparing to unpack .../figlet_2.2.5-2_armhf.deb ...
Unpacking figlet (2.2.5-2) ...
Setting up figlet (2.2.5-2) ...
update-alternatives: using /usr/bin/figlet-figlet to provide /usr/bin/figlet (figlet) in
auto mode
Processing triggers for man-db (2.7.6.1-2) ...
```

Sau khi quá trình tải xuống hoàn tất, tất cả các tệp sẽ được sao chép vào các vị trí thích hợp, mọi cấu hình bổ sung sẽ được thực hiện và lệnh sẽ trở nên khả dụng:

```
$ figlet Awesome!

  _
 / \_      _____ _ _ _ _ _ _ _ _ _ _ | |
 / _ \ \ / \ / / _ \ / _ | / _ \ | ' ` _ \ / _ \ |
 / ___ \ V V / __/\_ \ ( ) | | | | | | | | | ___/_
 /_/   \ \ / \ / \_||_/\_ \ / | | | | | | \_ \ ( )
```

Trong các bản phân phối dựa trên các gói RPM, tác vụ tìm kiếm được thực hiện bằng cách sử dụng `yum search package_name` hoặc `dnf search package_name`. Giả sử bạn muốn hiển thị một số văn bản theo một cách thoải mái hơn, đính kèm một chú bò hoạt hình, nhưng bạn không chắc rằng gói nào có thể thực hiện tác vụ đó. Đối với các gói DEB, các lệnh tìm kiếm RPM có chấp nhận các thuật ngữ mô tả:

```
$ yum search speaking cow
```

```
Last metadata expiration check: 1:30:49 ago on Tue 23 Apr 2019 11:02:33 PM -03.
===== Name & Summary Matched: speaking, cow =====
cowsay.noarch : Configurable speaking/thinking cow
```

Sau khi tìm thấy một gói phù hợp tại kho lưu trữ, gói đó có thể được cài đặt bằng lệnh `yum install package_name` hoặc `dnf install package_name`:

```
$ sudo yum install cowsay
```

```
Last metadata expiration check: 2:41:02 ago on Tue 23 Apr 2019 11:02:33 PM -03.
Dependencies resolved.
=====
Package           Arch           Version           Repository         Size
=====
Installing:
cowsay            noarch         3.04-10.fc28     fedora             46 k

Transaction Summary
=====
Install 1 Package

Total download size: 46 k
Installed size: 76 k
Is this ok [y/N]: y
```

Một lần nữa, gói bạn muốn và tất cả các phần phụ thuộc có thể có của nó sẽ được tải xuống và cài đặt:

```
Downloading Packages:
cowsay-3.04-10.fc28.noarch.rpm          490 kB/s | 46 kB    00:00
=====
Total                                  53 kB/s | 46 kB    00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing      :                               1/1
Installing    : cowsay-3.04-10.fc28.noarch  1/1
Running scriptlet: cowsay-3.04-10.fc28.noarch 1/1
Verifying     : cowsay-3.04-10.fc28.noarch  1/1
```

```
Installed:
cowsay.noarch 3.04-10.fc28

Complete!
```

Lệnh `cowsay` (chú bò nói) sẽ thực hiện đúng như tên gọi của nó:

```
$ cowsay "Brought to you by yum"
_____
< Brought to you by yum >
-----
  \  ^__^
   \ (oo)\_______
      (__)\       )\/\
         ||----w |
         ||     ||
```

Mặc dù có thể không hữu dụng nhưng các lệnh `figlet` và `cowsay` là một cách để thu hút sự chú ý của những người dùng khác tới những thông tin cần thiết.

Gỡ bỏ Gói

Các lệnh được sử dụng để cài đặt gói cũng là các lệnh được sử dụng để gỡ chúng. Tất cả các lệnh đều chấp nhận từ khóa `remove` để gỡ cài đặt gói: `apt-get remove package_name` hoặc `apt remove package_name` đối với các gói DEB và `yum remove package_name` hoặc `dnf remove package_name` đối với các gói RPM. Để thực hiện việc gỡ bỏ gói, ta cũng cần đến lệnh `sudo`. Ví dụ: để xóa gói `figlet` đã cài đặt trước đó khỏi bản phân phối dựa trên DEB:

```
$ sudo apt-get remove figlet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  figlet
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 741 kB disk space will be freed.
Do you want to continue? [Y/n] Y
```

Sau khi xác nhận lệnh, gói sẽ bị xóa khỏi hệ thống:


```
(Reading database ... 115775 files and directories currently installed.)
Removing figlet (2.2.5-2) ...
Processing triggers for man-db (2.7.6.1-2) ...
```

Quy trình cũng sẽ tương tự như vậy trên hệ thống dựa trên RPM. Ví dụ: để xóa gói *cowsay* đã cài đặt trước đó khỏi bản phân phối dựa trên RPM:

```
$ sudo yum remove cowsay
Dependencies resolved.
=====
Package            Arch             Version          Repository        Size
=====
Removing:
cowsay             noarch           3.04-10.fc28    @fedora           76 k

Transaction Summary
=====
Remove 1 Package

Freed space: 76 k
Is this ok [y/N]: y
```

Tương tự như vậy, hệ thống sẽ yêu cầu xác nhận và sau đó gói sẽ bị xóa khỏi hệ thống:

```
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :                                1/1
  Erasing       : cowsay-3.04-10.fc28.noarch     1/1
  Running scriptlet: cowsay-3.04-10.fc28.noarch  1/1
  Verifying     : cowsay-3.04-10.fc28.noarch     1/1

Removed:
  cowsay.noarch 3.04-10.fc28

Complete!
```

Các tệp cấu hình của các gói đã bị xóa vẫn sẽ được lưu giữ trên hệ thống, vì vậy chúng có thể được tái sử dụng nếu gói được cài đặt lại trong tương lai.

Ứng dụng Văn phòng

Các ứng dụng văn phòng được sử dụng để chỉnh sửa các tệp như văn bản, bản trình chiếu, bảng tính và các định dạng khác thường được sử dụng trong môi trường văn phòng. Các ứng dụng này thường được bao gồm trong các bộ sưu tập có tên *office suites*.

Trong một thời gian dài, bộ ứng dụng văn phòng được sử dụng nhiều nhất trong Linux là bộ *OpenOffice.org*. OpenOffice.org là phiên bản mã nguồn mở của *StarOffice suite* do Sun Microsystems phát hành. Vài năm sau đó, Sun đã được Oracle Corporation mua lại, từ đó dự án được chuyển qua cho Apache Foundation và OpenOffice.org được đổi tên thành *Apache OpenOffice*. Trong khi đó, một bộ ứng dụng văn phòng khác dựa trên cùng một mã nguồn cũng đã được phát hành bởi Document Foundation và được đặt tên là *LibreOffice*.

Hai bộ ứng dụng này có các tính năng cơ bản giống nhau và tương thích với các định dạng tài liệu đến từ *Microsoft Office*. Tuy nhiên, định dạng tài liệu được ưu tiên là *Định dạng Tài liệu Mở (Open Document Format)* - một định dạng tệp chuẩn ISO và hoàn toàn mở. Việc sử dụng các tệp ODF đảm bảo được việc các tài liệu có thể được chuyển qua lại giữa các hệ điều hành và ứng dụng từ các nhà cung cấp khác nhau, chẳng hạn như Microsoft Office. Các ứng dụng chính do OpenOffice/LibreOffice cung cấp là:

Writer

Trình soạn thảo văn bản

Calc

Trình bảng tính

Impress

Trình chiếu

Draw

Trình vẽ vec tơ

Math

Trình công thức toán học

Base

Trình Cơ sở dữ liệu

Cả LibreOffice và Apache OpenOffice đều là phần mềm mã nguồn mở, nhưng LibreOffice được cấp phép theo Giấy phép LGPLv3 và Apache OpenOffice được cấp phép theo Giấy phép Apache 2.0. Sự khác biệt về giấy phép ám chỉ rằng LibreOffice có thể kết hợp các cải tiến do Apache

OpenOffice thực hiện, nhưng Apache OpenOffice lại không thể kết hợp các cải tiến do LibreOffice thực hiện. Cộng với một cộng đồng các nhà phát triển năng nổ hơn, đây chính là lý do khiến hầu hết các bản phân phối đều sử dụng LibreOffice làm bộ ứng dụng văn phòng mặc định của chúng.

Trình duyệt Web

Đối với phần lớn người dùng, mục đích chính của máy tính là cung cấp quyền truy cập Internet. Ngày nay, các trang web có thể hoạt động như một ứng dụng với đầy đủ các tính năng, cùng ưu điểm là có thể truy cập từ mọi nơi mà không cần cài đặt thêm các phần mềm khác. Điều này khiến trình duyệt web trở thành ứng dụng quan trọng nhất của hệ điều hành, ít nhất là đối với người dùng phổ thông.

TIP

Một trong những nguồn bổ ích nhất để tìm hiểu về phát triển trang mạng là MDN Web Docs, có sẵn tại <https://developer.mozilla.org/>. MDN Web Docs được duy trì bởi Mozilla và có đầy đủ các hướng dẫn dành cho người mới bắt đầu, cũng như tài liệu tham khảo về hầu hết các công nghệ web hiện đại.

Các trình duyệt web chính trong môi trường Linux là *Google Chrome* và *Mozilla Firefox*. Chrome là trình duyệt web do Google quản lý, nhưng nó lại được dựa trên một trình duyệt mã nguồn mở có tên *Chromium*; trình duyệt này có thể được cài đặt bằng trình quản lý gói của bản phân phối và hoàn toàn tương thích với Chrome. Firefox được duy trì bởi Mozilla - một tổ chức phi lợi nhuận - và là trình duyệt có nguồn gốc được liên kết với Netscape - trình duyệt web phổ biến đầu tiên áp dụng mô hình mã nguồn mở. Quỹ Mozilla (The Mozilla Foundation) có liên quan mật thiết tới việc phát triển các tiêu chuẩn mở thiết lập nền tảng cho trang mạng hiện đại.

Mozilla cũng phát triển các ứng dụng khác như ứng dụng e-mail *Thunderbird*. Nhiều người dùng lựa chọn sử dụng webmail thay vì ứng dụng email chuyên dụng, nhưng một ứng dụng khác như Thunderbird lại có thể cung cấp các tính năng bổ sung và tích hợp tốt nhất với các ứng dụng khác trên máy tính để bàn.

Truyền thông Đa phương tiện

So với các ứng dụng web có sẵn, các ứng dụng dành cho máy tính để bàn vẫn là lựa chọn tốt nhất để tạo các nội dung đa phương tiện. Các hoạt động liên quan đến đa phương tiện như kết xuất video thường yêu cầu lượng tài nguyên hệ thống cao, vì thế nên chúng sẽ được quản lý tốt nhất bởi một ứng dụng cục bộ trên máy tính để bàn. Một số ứng dụng đa phương tiện phổ biến nhất cho môi trường Linux và cách sử dụng chúng sẽ được liệt kê dưới đây.

Blender

Một trình kết xuất 3D để làm phim hoạt hình. Blender cũng có thể được sử dụng để xuất các vật thể 3D được in bằng máy in 3D.

GIMP

Một trình chỉnh sửa hình ảnh với đầy đủ các tính năng sánh ngang với *Adobe Photoshop*, nhưng cũng lại có các khái niệm và công cụ riêng để làm việc với hình ảnh. GIMP có thể được sử dụng để tạo, chỉnh sửa và lưu hầu hết các tệp bitmap như JPEG, PNG, GIF, TIFF và nhiều dạng khác.

Inkscape

Một trình chỉnh sửa đồ họa vector, tương tự như *Corel Draw* hoặc *Adobe Illustrator*. Định dạng mặc định của Inkscape là SVG - đây cũng là một tiêu chuẩn mở cho đồ họa vector. Các tệp SVG có thể được mở bởi bất kỳ trình duyệt web nào và do bản chất là đồ họa véc-tơ nên nó có thể được sử dụng trong bố cục linh hoạt của các trang web.

Audacity

Một trình chỉnh sửa âm thanh. Audacity có thể được sử dụng để lọc, áp hiệu ứng và chuyển đổi giữa nhiều định dạng âm thanh khác nhau như MP3, WAV, OGG, FLAC, v.v.

ImageMagick

Một công cụ dòng lệnh dùng để chuyển đổi và chỉnh sửa hầu hết các loại tệp hình ảnh. Nó cũng có thể được sử dụng để tạo tài liệu PDF từ các tệp hình ảnh và ngược lại. Ngoài ra còn có nhiều ứng dụng dành riêng cho việc phát đa phương tiện. Ứng dụng phổ biến nhất để trình phát video là *VLC*, nhưng một số người dùng lại thích các lựa chọn thay thế khác hơn, chẳng hạn như *smplyer*. Trình phát nhạc cục bộ cũng có nhiều tùy chọn như *Audacious*, *Banshee* hay *Amarok*; các trình phát này cũng có thể quản lý một tập hợp các tệp âm thanh cục bộ.

Chương trình Máy chủ

Khi trình duyệt web tải một trang từ một trang web, thực tế là nó sẽ kết nối với một máy tính từ xa và yêu cầu một thông tin cụ thể. Trong trường hợp đó, máy tính chạy trình duyệt web được gọi là *máy khách*, và máy tính từ xa được gọi là *máy chủ*.

Máy chủ - có thể là một máy tính để bàn thông thường hoặc một phần cứng chuyên dụng - cần một chương trình đặc thù để quản lý từng loại thông tin mà nó sẽ cung cấp. Về việc phục vụ các trang web, hầu hết các máy chủ trên thế giới đều triển khai các chương trình máy chủ mã nguồn mở. Chương trình máy chủ cụ thể này được gọi là *Máy chủ HTTP* (HTTP là viết tắt của *Hyper Text Transfer Protocol* - Giao thức truyền siêu văn bản) và những chương trình phổ biến nhất là *Apache*, *Nginx* và *lighttpd*.

Ngay cả các trang web đơn giản cũng có thể cần nhiều yêu cầu; các yêu cầu đó có thể là các tệp thông thường — được gọi là nội dung tĩnh — hoặc nội dung động được kết xuất từ nhiều nguồn khác nhau. Vai trò của máy chủ HTTP là thu thập và gửi tất cả dữ liệu được yêu cầu trở lại trình

duyet, sau đó trình duyệt sẽ sắp xếp nội dung như được xác định bởi tài liệu HTML nhận được (HTML là viết tắt của *Hyper Text Markup Language* - Ngôn ngữ đánh dấu siêu văn bản) và các tệp hỗ trợ khác. Do đó, việc hiển thị trang web sẽ bao gồm các hoạt động được thực hiện ở cả phía máy chủ và phía máy khách. Cả hai bên có thể sử dụng các tệp lệnh tùy chỉnh để hoàn thành các tác vụ cụ thể. Về phía máy chủ HTTP, việc sử dụng ngôn ngữ kịch lệnh PHP là khá phổ biến. JavaScript là ngôn ngữ lệnh được sử dụng ở phía máy khách (trình duyệt web).

Các chương trình máy chủ có thể cung cấp nhiều các loại thông tin khác nhau. Việc chương trình máy chủ yêu cầu thông tin được cung cấp bởi các chương trình máy chủ khác không phải là hiếm. Đó là trường hợp khi máy chủ HTTP yêu cầu thông tin được cung cấp bởi máy chủ cơ sở dữ liệu.

Chẳng hạn như khi một trang web động được yêu cầu, máy chủ HTTP thường truy vấn cơ sở dữ liệu để thu thập tất cả các phần thông tin cần thiết và gửi nội dung động này trở lại máy khách. Theo cách tương tự, khi người dùng đăng ký trên một trang web, máy chủ HTTP sẽ thu thập dữ liệu do khách hàng gửi và lưu trữ nó trong cơ sở dữ liệu.

Cơ sở dữ liệu là một tập hợp thông tin có tổ chức. Một máy chủ cơ sở dữ liệu lưu trữ nội dung theo kiểu định dạng nội dung; việc này giúp nó có thể đọc, ghi và liên kết một lượng lớn dữ liệu một cách đáng tin cậy ở tốc độ cao. Máy chủ cơ sở dữ liệu mã nguồn mở được sử dụng trong nhiều ứng dụng chứ không chỉ trên Internet. Ngay cả các ứng dụng cục bộ cũng có thể lưu trữ dữ liệu bằng cách kết nối với máy chủ cơ sở dữ liệu cục bộ. Loại cơ sở dữ liệu phổ biến nhất là *cơ sở dữ liệu quan hệ*, trong đó dữ liệu được tổ chức trong các bảng đã được xác định trước. Các cơ sở dữ liệu quan hệ mã nguồn mở phổ biến nhất là *MariaDB* (có nguồn gốc từ *MySQL*) và *PostgreSQL*.

Chia sẻ Dữ liệu

Trong các mạng cục bộ, ví dụ như các mạng trong văn phòng và gia đình, các máy tính được kỳ vọng là không chỉ để truy cập Internet mà còn có thể giao tiếp với nhau. Đôi khi một máy tính có thể đóng vai trò là máy chủ, đôi khi chính nó lại đóng vai trò là máy khách. Điều này là cần thiết khi một người muốn truy cập các tệp trên một máy tính khác ở trong cùng một mạng — ví dụ: truy cập một tệp được lưu trữ trên máy tính để bàn từ một thiết bị di động — mà không phải làm tác vụ rắc rối là sao chép tệp đó vào ổ USB hoặc các thiết bị tương tự.

Trong các máy Linux, *NFS (Network File System - Hệ thống Tệp Mạng)* thường được sử dụng khá phổ biến. Giao thức NFS là cách tiêu chuẩn để chia sẻ các hệ thống tệp trong các mạng chỉ được trang bị với máy Unix/Linux. Với NFS, một máy tính có thể chia sẻ một hoặc nhiều thư mục của nó với các máy tính nhất định cùng ở trong mạng để chúng có thể đọc và ghi trên tệp trong các thư mục này. NFS thậm chí có thể được sử dụng để chia sẻ toàn bộ cây thư mục của hệ điều hành với các máy khách sẽ sử dụng cây thư mục đó để khởi động. Những máy tính này được gọi là *máy khách có cấu hình tối thiểu* (thin client) và thường được sử dụng trong các mạng lớn để tránh việc bảo trì từng hệ điều hành riêng lẻ trên mỗi máy.

Nếu có các loại hệ điều hành khác được gắn với mạng thì người dùng được khuyến cáo sử dụng giao thức chia sẻ dữ liệu mà tất cả các hệ điều hành đều có thể hiểu được. *Samba* có thể đáp ứng được yêu cầu này. Samba, ban đầu được dành cho hệ điều hành Windows, triển khai giao thức chia sẻ tệp qua mạng; ngày nay, nó tương thích với tất cả các hệ điều hành lớn. Với Samba, không chỉ các máy tính trong mạng nội bộ có thể chia sẻ tệp mà ngay cả máy in cũng có thể.

Trên một số mạng cục bộ, quyền đăng nhập trên máy trạm được cấp bởi *máy chủ trung tâm*; máy chủ này quản lý quyền truy cập vào các tài nguyên cục bộ và từ xa khác nhau. Máy chủ trung tâm là một dịch vụ được cung cấp bởi *Active Directory* của Microsoft. Máy trạm Linux có thể liên kết với máy chủ trung tâm bằng cách sử dụng Samba hoặc hệ thống con xác thực có tên là *SSSD*. Kể từ phiên bản 4, Samba cũng có thể hoạt động như một máy chủ trung tâm trên các mạng không đồng nhất.

Nếu mục tiêu ở đây là triển khai một giải pháp điện toán đám mây có khả năng cung cấp nhiều phương pháp chia sẻ dữ liệu dựa trên web khác nhau thì ta có thể xem xét hai lựa chọn là *ownCloud* và *Nextcloud*. Hai dự án này rất giống nhau vì Nextcloud chính là một sản phẩm nhánh của ownCloud. Điều này không có gì lạ đối với các dự án mã nguồn mở; các sản phẩm nhánh như vậy thường được gọi là *fork*. Cả hai đều cung cấp các tính năng cơ bản giống nhau: chia sẻ và đồng bộ hóa tệp, không gian làm việc cộng tác, lịch, danh bạ và thư; tất cả đều thông qua giao diện máy tính để bàn, thiết bị di động và web. Nextcloud cũng cung cấp hội nghị truyền hình/âm thanh riêng tư, trong khi ownCloud tập trung hơn vào việc chia sẻ tệp và tích hợp với các phần mềm của bên thứ ba. Nhiều tính năng khác cũng được cung cấp dưới dạng tiện ích bổ sung có thể được kích hoạt sau khi cần.

Cả ownCloud và Nextcloud đều cung cấp phiên bản trả phí với các tính năng bổ sung và hỗ trợ mở rộng. Điều làm cho chúng khác biệt với các giải pháp thương mại khác là người dùng có thể cài đặt miễn phí Nextcloud hoặc ownCloud trên máy chủ riêng giúp tránh việc lưu giữ dữ liệu nhạy cảm trên máy chủ không xác định. Vì tất cả các dịch vụ đều phụ thuộc vào giao tiếp HTTP và được viết bằng PHP nên quá trình cài đặt phải được thực hiện trên một máy chủ trang mạng được cấu hình sẵn, chẳng hạn như Apache. Nếu bạn muốn cài đặt ownCloud hoặc Nextcloud trên máy chủ của riêng mình, hãy đảm bảo bạn có kích hoạt HTTPS để mã hóa tất cả kết nối với đám mây của bạn.

Quản Trị Mạng

Giao tiếp giữa các máy tính chỉ có thể thực hiện được nếu mạng hoạt động một cách chính xác. Thông thường, cấu hình mạng được thực hiện bởi một tập hợp các chương trình chạy trên bộ định tuyến; nó chịu trách nhiệm thiết lập và kiểm tra tính khả dụng của mạng. Để đạt được điều này, người ta sử dụng hai dịch vụ mạng cơ bản là *DHCP* (*Dynamic Host Configuration Protocol* - Giao thức cấu hình máy chủ động) và *DNS* (*Domain Name System* - Hệ Thống Tên Miền).

DHCP chịu trách nhiệm gán địa chỉ IP cho máy chủ khi cáp mạng được kết nối hoặc khi thiết bị

truy cập vào mạng không dây. Khi kết nối Internet, máy chủ DHCP của ISP sẽ cung cấp địa chỉ IP cho thiết bị đang yêu cầu. Máy chủ DHCP cũng rất hữu ích trong các mạng cục bộ trong việc tự động cung cấp địa chỉ IP cho tất cả các thiết bị được kết nối. Nếu DHCP không được định cấu hình hoặc hoạt động không bình thường thì địa chỉ IP của từng thiết bị được kết nối với mạng phải được định cấu hình theo cách thủ công. Việc thiết lập địa chỉ IP theo cách thủ công trên các mạng lớn hoặc thậm chí trong các mạng nhỏ là không thực tế, đó là lý do tại sao hầu hết các bộ định tuyến mạng đều đi kèm với máy chủ DHCP đã được cấu hình sẵn theo mặc định.

Cần có địa chỉ IP để liên lạc với một thiết bị khác trên một mạng IP, nhưng các tên miền như `www.lpi.org` thường sẽ dễ nhớ hơn một dãy số IP như `203.0.113.165`. Tuy vậy, bản thân tên miền không đủ để thiết lập liên lạc qua mạng. Đó là lý do tại sao tên miền cần được máy chủ DNS dịch sang địa chỉ IP. Địa chỉ IP của máy chủ DNS được cung cấp bởi máy chủ DHCP của ISP và nó sẽ được sử dụng bởi tất cả các hệ thống được kết nối để dịch tên miền thành địa chỉ IP. Cả cài đặt DHCP và DNS đều có thể được thay đổi bằng cách truy cập vào giao diện web do bộ định tuyến cung cấp. Chẳng hạn, ta có thể hạn chế chỉ gán IP cho các thiết bị đã biết hoặc liên kết một địa chỉ IP cố định cho các máy cụ thể. Máy chủ DNS mặc định do ISP cung cấp cũng có thể được thay đổi. Một số máy chủ DNS của bên thứ ba (chẳng hạn như máy chủ do Google hoặc OpenDNS cung cấp) đôi khi có thể cho phản hồi nhanh hơn và có thêm nhiều các tính năng bổ sung.

Ngôn ngữ Lập trình

Tất cả các chương trình máy tính (chương trình máy khách và máy chủ, ứng dụng máy tính để bàn và chính hệ điều hành) đều được tạo ra bởi một hoặc nhiều ngôn ngữ lập trình. Các chương trình có thể là một tệp đơn lẻ hoặc một hệ thống phức tạp gồm hàng trăm tệp mà hệ điều hành coi là một chuỗi lệnh để bộ xử lý và các thiết bị khác diễn giải và thực hiện.

Có rất nhiều ngôn ngữ lập trình dành cho từng mục đích khác nhau, và hệ thống Linux cung cấp đa dạng các loại ngôn ngữ này. Vì phần mềm mã nguồn mở cũng bao gồm nguồn của các chương trình nên các hệ thống Linux cũng cung cấp cho các nhà phát triển những điều kiện hoàn hảo để hiểu, sửa đổi hoặc tạo ra phần mềm theo nhu cầu của riêng họ.

Mỗi một chương trình đều xuất phát dưới dạng một tệp văn bản gọi là *mã nguồn*. Mã nguồn này được viết bằng một ngôn ngữ ít nhiều thân thiện với con người để mô tả chương trình đang làm gì. Bộ xử lý máy tính không thể thực thi trực tiếp mã nguồn này. Trong *ngôn ngữ biên dịch*, nó phải được chuyển đổi thành *tệp nhị phân* để máy tính có thể thực thi được. Một chương trình có tên *trình biên dịch* sẽ chịu trách nhiệm thực hiện chuyển đổi mã nguồn sang dạng có thể thực thi được. Vì mỗi tệp nhị phân đã qua biên dịch sẽ chỉ dành riêng cho một loại bộ xử lý cụ thể, chương trình này có thể phải được biên dịch lại để chạy trên một loại máy tính khác.

Trong *ngôn ngữ phiên dịch*, chương trình không cần phải được biên dịch trước. Thay vào đó, một *trình phiên dịch* sẽ đọc mã nguồn và thực hiện luôn lệnh của nó mỗi khi chương trình được chạy.

Điều này làm cho việc phát triển trở nên dễ dàng và nhanh chóng hơn, nhưng đồng thời các chương trình được biên dịch lại có xu hướng chạy chậm hơn các chương trình được biên dịch.

Một số ngôn ngữ lập trình phổ biến nhất:

JavaScript

JavaScript là ngôn ngữ lập trình chủ yếu được sử dụng trong các trang web. Các ứng dụng JavaScript ban đầu đều rất đơn giản, chẳng hạn như các quy trình xác thực biểu mẫu. Ngày nay, JavaScript được coi là ngôn ngữ hạng nhất và được sử dụng để tạo các ứng dụng rất phức tạp không chỉ trên web mà còn trên máy chủ và thiết bị di động.

C

Ngôn ngữ lập trình C có liên quan chặt chẽ với các hệ điều hành, đặc biệt là Unix, nhưng nó có thể được sử dụng để viết bất kỳ loại chương trình nào cho hầu hết mọi loại thiết bị. Ưu điểm lớn của C là tính linh hoạt và tốc độ của nó. Cùng một mã nguồn có thể được biên dịch để chạy trong các nền tảng và hệ điều hành khác nhau mà không cần sửa đổi (hoặc sửa đổi rất ít) nếu được viết bằng C. Tuy nhiên, sau khi được biên dịch, chương trình sẽ chỉ chạy được trong hệ thống mục tiêu.

Java

Điểm mạnh chính của Java là các chương trình được viết bằng ngôn ngữ này rất tiện dụng, có nghĩa là cùng một chương trình có thể được thực thi trong nhiều hệ điều hành khác nhau. Dù tên là Java nhưng ngôn ngữ này không hề liên quan đến JavaScript.

Perl

Perl là ngôn ngữ lập trình được sử dụng nhiều nhất để xử lý nội dung văn bản. Perl nhấn mạnh vào các biểu thức chính quy, khiến nó trở thành một ngôn ngữ phù hợp để lọc và phân tích văn bản.

Shell

Shell, đặc biệt là Bash shell, không chỉ là một ngôn ngữ lập trình mà còn là một giao diện tương tác để chạy các chương trình khác. Các chương trình vỏ, hay còn được gọi là *tệp lệnh vỏ*, có thể tự động hóa các tác vụ phức tạp hoặc lặp đi lặp lại trên môi trường dòng lệnh.

Python

Python là ngôn ngữ lập trình rất phổ biến đối với sinh viên và các chuyên gia không tham gia trực tiếp vào khoa học máy tính. Mặc dù cũng có các tính năng nâng cao, Python vẫn là một phương thức tốt để bắt đầu học lập trình vì cách tiếp cận dễ sử dụng của nó.

PHP

PHP được sử dụng nhiều nhất làm ngôn ngữ tệp lệnh ở phía máy chủ để tạo nội dung cho web.

Hầu hết các trang HTML trực tuyến đều không phải là các tệp tĩnh mà là các nội dung động do máy chủ tạo từ nhiều nguồn khác nhau, chẳng hạn như cơ sở dữ liệu. Các chương trình PHP — đôi khi chỉ được gọi là các trang PHP hoặc tệp lệnh PHP — thường được sử dụng để tạo loại nội dung này. Thuật ngữ LAMP xuất phát từ sự kết hợp của hệ điều hành Linux, máy chủ HTTP Apache, cơ sở dữ liệu MySQL (hoặc MariaDB) và lập trình PHP. Máy chủ LAMP là một giải pháp rất phổ biến để chạy các máy chủ web. Ngoài PHP, tất cả các ngôn ngữ lập trình đã nêu trên cũng có thể được sử dụng để triển khai các ứng dụng tương tự.

C và Java đều là ngôn ngữ biên dịch. Để được hệ thống thực thi, mã nguồn viết bằng C sẽ được chuyển đổi thành mã máy nhị phân, trong khi mã nguồn Java sẽ được chuyển đổi thành *bytecode* được thực thi trong môi trường phần mềm đặc biệt có tên *Máy ảo Java*. JavaScript, Perl, Shell script, Python và PHP đều là các ngôn ngữ thông dịch hay còn được gọi là *ngôn ngữ tệp lệnh*.

Bài tập Hướng dẫn

1. Hãy xác định xem mỗi dòng lệnh sau đây có liên quan đến *Hệ thống gói Debian* hay *Hệ thống gói Red Hat* hay không:

dpkg	
rpm	
apt-get	
yum	
dnf	

2. Lệnh nào có thể được sử dụng để cài đặt Blender trên Ubuntu? Sau khi cài đặt, làm thế nào để chạy chương trình?

3. Ứng dụng nào từ LibreOffice suite có thể được sử dụng để làm việc với bảng tính điện tử?

4. Trình duyệt web mã nguồn mở nào được sử dụng làm cơ sở cho sự phát triển của Google Chrome?

5. SVG là một tiêu chuẩn mở cho đồ họa vector. Ứng dụng phổ biến nhất để chỉnh sửa tệp SVG trong hệ thống Linux là gì?

6. Đối với mỗi định dạng tệp sau đây, hãy cho biết tên ứng dụng tương ứng có thể mở và chỉnh sửa tệp đó:

png	
doc	
xls	
ppt	
wav	

7. Gói phần mềm nào cho phép chia sẻ tệp giữa các máy Linux và Windows qua mạng cục bộ?

Bài tập Mở rộng

1. Bạn đã biết rằng các tệp cấu hình được lưu giữ ngay cả khi gói của nó bị xóa khỏi hệ thống. Làm cách nào để có thể tự động xóa gói có tên *cups* và các tệp cấu hình của nó khỏi hệ thống dựa trên DEB?

2. Giả sử bạn có nhiều tệp ảnh TIFF và muốn chuyển chúng sang định dạng JPEG. Gói phần mềm nào có thể được sử dụng để chuyển đổi các tệp đó trực tiếp tại dòng lệnh?

3. Cần cài đặt gói phần mềm nào để có thể mở tài liệu Microsoft Word do người dùng Windows gửi cho bạn?

4. Hàng năm, [linuxquestions.org](https://www.linuxquestions.org) sẽ thực hiện một cuộc khảo sát về các ứng dụng Linux phổ biến nhất. Truy cập <https://www.linuxquestions.org/questions/2018-linuxquestions-org-members-choice-awards-128/> và tìm hiểu xem ứng dụng máy tính để bàn nào là phổ biến nhất đối với những người dùng Linux có kinh nghiệm.

Tóm tắt

Trong bài học này, bạn đã học về:

- Các hệ thống quản lý gói được sử dụng trong các bản phân phối Linux chính
- Các ứng dụng mã nguồn mở có thể chỉnh sửa các định dạng tệp phổ biến
- Các chương trình máy chủ đứng sau nhiều dịch vụ mạng cục bộ và Internet quan trọng
- Các ngôn ngữ lập trình phổ biến và công dụng của chúng

Đáp án Bài tập Hướng dẫn

1. Hãy xác định xem mỗi dòng lệnh sau đây có liên quan đến *Hệ thống gói Debian* hay *Hệ thống gói Red Hat* hay không:

<code>dpkg</code>	Debian packaging system
<code>rpm</code>	Red Hat packaging system
<code>apt-get</code>	Debian packaging system
<code>yum</code>	Red Hat packaging system
<code>dnf</code>	Red Hat packaging system

2. Lệnh nào có thể được sử dụng để cài đặt Blender trên Ubuntu? Sau khi cài đặt, làm thế nào để chạy chương trình?

Lệnh `apt-get install blender`. Tên gói phải được chỉ định bằng chữ thường. Chương trình có thể chạy trực tiếp trên cửa sổ dòng lệnh bằng lệnh `blender` hoặc bằng cách chọn nó trên menu ứng dụng.

3. Ứng dụng nào từ LibreOffice suite có thể được sử dụng để làm việc với bảng tính điện tử?

Calc

4. Trình duyệt web nguồn mở nào được sử dụng làm cơ sở cho sự phát triển của Google Chrome?

Chromium

5. SVG là một tiêu chuẩn mở cho đồ họa vector. Ứng dụng phổ biến nhất để chỉnh sửa tệp SVG trong hệ thống Linux là gì?

Inkscape

6. Đối với mỗi định dạng tệp sau đây, hãy cho biết tên ứng dụng tương ứng có thể mở và chỉnh sửa tệp đó:

<code>png</code>	Gimp
<code>doc</code>	LibreOffice Writer
<code>xls</code>	LibreOffice Calc
<code>ppt</code>	LibreOffice Impress
<code>wav</code>	Audacity

7. Gói phần mềm nào cho phép chia sẻ tệp giữa các máy Linux và Windows qua mạng cục bộ?

Samba

Đáp án Bài tập Mở rộng

1. Bạn đã biết rằng các tệp cấu hình được lưu giữ ngay cả khi gói của nó bị xóa khỏi hệ thống. Làm cách nào để có thể tự động xóa gói có tên *cups* và các tệp cấu hình của nó khỏi hệ thống dựa trên DEB?

```
apt-get purge cups
```

2. Giả sử bạn có nhiều tệp ảnh TIFF và muốn chuyển chúng sang định dạng JPEG. Gói phần mềm nào có thể được sử dụng để chuyển đổi các tệp đó trực tiếp tại dòng lệnh?

ImageMagick

3. Cần cài đặt gói phần mềm nào để có thể mở tài liệu Microsoft Word do người dùng Windows gửi cho bạn?

LibreOffice hoặc OpenOffice

4. Hàng năm, [linuxquestions.org](https://www.linuxquestions.org) sẽ thực hiện một cuộc khảo sát về các ứng dụng Linux phổ biến nhất. Truy cập <https://www.linuxquestions.org/questions/2018-linuxquestions-org-members-choice-awards-128/> và tìm hiểu xem ứng dụng máy tính để bàn nào là phổ biến nhất đối với những người dùng Linux có kinh nghiệm.

Trình duyệt: Firefox. Trình duyệt mail: Thunderbird. Trình phát đa phương tiện: VLC. Trình chỉnh sửa đồ họa raster: GIMP.



**Linux
Professional
Institute**

1.3 Phần mềm Mã Nguồn Mở và Giấy phép

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 1.3](#)

Khối lượng

1

Các lĩnh vực kiến thức chính

- Triết lý Mã Nguồn Mở
- Cấp phép cho Mã Nguồn Mở
- Tổ chức Phần mềm Tự do / Free Software Foundation (FSF), Sáng kiến Mã Nguồn Mở / Open Source Initiative (OSI)

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

- Copyleft, Permissive
- GPL, BSD, Creative Commons
- Free Software, Open Source Software, FOSS, FLOSS
- Open source business models



1.3 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	1 Cộng đồng Linux và Sự nghiệp trong lĩnh vực Mã nguồn mở
Mục tiêu:	1.3 Phần mềm Mã nguồn mở và vấn đề Cấp phép
Bài:	1 trên 1

Giới thiệu

Trong khi các thuật ngữ *phần mềm tự do* và *phần mềm mã nguồn mở* đã được sử dụng rộng rãi, một số hiểu lầm về ý nghĩa của chúng vẫn còn tồn tại. Đặc biệt, khái niệm “tự do” cần được xem xét kỹ hơn. Hãy cùng bắt đầu với định nghĩa của hai thuật ngữ này.

Định nghĩa về Phần mềm Mã nguồn Mở và Tự do

Tiêu chí của Phần mềm Tự do

Trước hết, “tự do” (free) trong ngữ cảnh của phần mềm tự do không liên quan gì đến “miễn phí” (free), hay như người sáng lập *Quy Phần mềm Tự do* (FSF) Richard Stallman đã nói một cách ngắn gọn:

"Để hiểu rõ khái niệm này, chúng ta có thể coi “tự do” như trong “tự do ngôn luận” chứ không phải trong “` vào cổng tự do `*”. — Richard Stallman, Phần mềm Tự do là gì?

Bản gốc là "To understand the concept, you should think of "free" as in "free speech," not as in "free beer". Bản dịch đã được thay đổi để phù hợp với văn phong và nhận thức trong tiếng Việt.

Bất kể việc người dùng có phải trả tiền cho phần mềm hay không, có bốn tiêu chí để cấu thành phần mềm tự do. Richard Stallman đã mô tả những tiêu chí này là "bốn quyền tự do thiết yếu" và ông đếm chúng từ số 0:

- "Quyền tự do chạy chương trình theo ý muốn cho mọi mục đích (Quyền tự do số 0)."

Phần mềm được sử dụng ở đâu, như thế nào và cho mục đích gì không bị bắt buộc tuân theo quy định hoặc hạn chế.

- "Quyền tự do nghiên cứu cách thức hoạt động của chương trình và thay đổi nó để chương trình hoạt động theo ý muốn của mình (Quyền tự do số 1). Truy cập vào mã nguồn là điều kiện tiên quyết cho điều này."

Ai cũng có thể thay đổi phần mềm theo ý tưởng và nhu cầu của họ. Ngược lại, điều này ám chỉ rằng cái gọi là *mã nguồn* (tức là tất cả các tệp chứa trong phần mềm) phải có sẵn ở dạng mà người lập trình có thể đọc được. Và tất nhiên, quyền này áp dụng cho một người dùng có thể muốn thêm một tính năng duy nhất cũng như cho các công ty phần mềm muốn xây dựng các hệ thống phức tạp như hệ điều hành điện thoại thông minh hoặc phần sụn của bộ định tuyến.

- "Quyền tự do phân phối lại các bản sao để giúp đỡ những người khác (Quyền tự do số 2)."

Quyền tự do này khuyến khích mạnh mẽ mỗi một người dùng chia sẻ phần mềm với những người khác. Do đó, trọng điểm ở đây là phân phối rộng nhất có thể và từ đó có được cộng đồng người dùng và nhà phát triển cũng phát triển rộng nhất có thể; cộng đồng này, trên cơ sở các quyền tự do này, sẽ tiếp tục phát triển và cải thiện phần mềm vì lợi ích của tất cả mọi người.

- "Quyền tự do phân phối bản sao của các phiên bản đã được sửa đổi của mình cho người khác (Quyền tự do số 3). Bằng cách này, chúng ta có thể tạo cơ hội cho cả cộng đồng hưởng lợi từ những thay đổi của mình. Quyền truy cập vào mã nguồn là điều kiện tiên quyết cho điều này."

Đây không chỉ là về vấn đề phân phối phần mềm tự do mà còn là về việc phân phối phần mềm tự do đã được sửa đổi. Bất kỳ ai thực hiện các thay đổi đối với phần mềm tự do đều có quyền cung cấp các thay đổi đó cho những người khác. Song song với đó, họ cũng phải có nghĩa vụ với quyền tự do của người khác, tức là họ không được hạn chế các quyền tự do ban đầu khi phân phối phần mềm ngay cả khi họ đã sửa đổi hoặc mở rộng phần mềm đó. Ví dụ: nếu một nhóm nhà phát triển có nhiều ý tưởng đa dạng về định hướng của một phần mềm cụ thể so với các tác giả ban đầu thì nhóm đó có thể tách nhánh phát triển của riêng mình (được gọi là *fork*) và tiếp tục phát triển nó như một dự án mới. Nhưng tất nhiên, tất cả các nghĩa vụ liên quan đến các quyền tự do này vẫn phải có.

Sự nhấn mạnh vào ý nghĩa của quyền tự do cũng nhất quán trong chừng mực mà mọi phong trào tự do đều cùng *chống lại* một loại đối tượng, cụ thể là một thể lực đàn áp các quyền tự do được mặc nhận và coi phần mềm là tài sản và muốn khoá nó lại. Ngược lại với phần mềm miễn phí, những phần mềm như vậy được gọi là *phần mềm độc quyền*.

Phần mềm Mã nguồn mở vs. Phần mềm Tự do

Đối với nhiều người, *phần mềm tự do* và *phần mềm mã nguồn mở* có ý nghĩa như nhau. Chữ viết tắt FOSS được sử dụng thường xuyên cho *Phần mềm Mã nguồn mở và Tự do* (Free and Open Source Software) nhấn mạnh vào điểm chung này. FLOSS là viết tắt của *Free/Libre and Open Source Software* (Phần mềm Tự do và Mã nguồn mở) là một thuật ngữ khác cũng khá phổ biến; nó nhấn mạnh vào khái niệm quyền tự do nhằm tránh nhầm lẫn về ngữ nghĩa cho các ngôn ngữ khác ngoài tiếng Anh. Tuy nhiên, sự khác biệt của hai thuật ngữ này xét trên khía cạnh nguồn gốc và sự phát triển của chúng cũng rất đáng để nhắc đến.

Thuật ngữ *phần mềm tự do* với định nghĩa về bốn quyền tự do được nêu trên đã được Richard Stallman và dự án GNU do ông thành lập vào năm 1985 đặt ra — gần 10 năm trước khi Linux xuất hiện. Cái tên “GNU không phải là Unix” (GNU is Not Unix) đã ngay lập tức cho thấy một ẩn ý: GNU bắt đầu như một sáng kiến nhằm phát triển một giải pháp thuyết phục về mặt kỹ thuật — giống như hệ điều hành Unix — từ đầu để cung cấp cho cộng đồng và liên tục cùng nhau cải thiện nó. Tính mở của mã nguồn chỉ đơn thuần là một sự cần thiết về mặt kỹ thuật và tổ chức, nhưng trong hình ảnh tự thân của nó, phong trào phần mềm tự do vẫn là một phong trào mang tính *xã hội và chính trị* — thậm chí theo một số người còn là phong trào ý thức hệ.

Với sự thành công của Linux, khả năng hợp tác của Internet cùng với hàng ngàn dự án và công ty nổi lên trong vũ trụ phần mềm mới này, khía cạnh xã hội ngày càng trở nên mờ nhạt. Bản thân tính mở của mã nguồn cũng thay đổi: từ một yêu cầu kỹ thuật trở thành một yếu tố mang tính định nghĩa: một khi mã nguồn được hiển thị, phần mềm lập tức đã được coi là “mã nguồn mở”. Các động cơ xã hội đã nhường chỗ cho một cách tiếp cận thực tế hơn trong việc phát triển phần mềm.

Phần mềm tự do và phần mềm mã nguồn mở có cùng một mục đích với cùng các phương pháp và ở trong cùng một cộng đồng toàn cầu gồm nhiều cá nhân, dự án và công ty. Nhưng vì chúng đến với nhau từ hai định hướng khác biệt - một bên là định hướng xã hội và một bên là định hướng thực dụng-kỹ thuật - nên đôi khi giữa hai bên sẽ xảy ra xung đột. Những xung đột này phát sinh khi kết quả của công việc chung không tương ứng với các mục tiêu ban đầu của cả hai phong trào. Điều này đặc biệt xảy ra khi phần mềm công khai mã nguồn của nó nhưng lại đồng thời không tôn trọng bốn quyền tự do của phần mềm tự do (chẳng hạn như khi xảy ra các hạn chế về việc tiết lộ, thay đổi hoặc kết nối với các thành phần phần mềm khác).

Giấy phép cho phép phần mềm khả dụng xác định các điều kiện mà phần mềm phải tuân theo liên

quan đến việc sử dụng, phân phối và sửa đổi. Và bởi vì các yêu cầu và động cơ có thể rất khác nhau nên vô số giấy phép khác nhau đã được tạo ra trong phạm vi của FOSS. Do cách tiếp cận cơ bản hơn hẳn, không có gì đáng ngạc nhiên khi phong trào phần mềm tự do không công nhận nhiều giấy phép mã nguồn mở là “tự do” và từ đó từ chối chúng. Ngược lại, điều này khó xảy ra đối với phần mềm mã nguồn mở do cách tiếp cận thực dụng hơn nhiều của nó.

Hãy cùng điếm qua những mặt vô cùng phức tạp của lĩnh vực giấy phép dưới đây.

Giấy phép

Không giống như tủ lạnh hay ô tô, phần mềm không phải là một sản phẩm *vật lý* mà là một sản phẩm *kỹ thuật số*. Do đó, trên thực tế, một công ty không thể chuyển quyền sở hữu một sản phẩm như vậy bằng cách bán nó và thay đổi quyền sở hữu vật lý—thay vào đó, họ sẽ chuyển giao quyền sử dụng sản phẩm đó và người dùng sẽ phải đồng thuận trên hợp đồng với các quyền sử dụng đó. Tất cả những quyền sử dụng và giới hạn *không* được sử dụng đều sẽ được ghi lại trong giấy phép phần mềm và từ đó chúng ta có thể hiểu được tầm quan trọng của chúng.

Trong khi các nhà cung cấp phần mềm sở hữu độc quyền lớn (chẳng hạn như Microsoft hoặc SAP) có giấy phép riêng được thiết kế tỉ mỉ cho từng sản phẩm của họ thì những người ủng hộ phần mềm tự do và mã nguồn mở ngay từ đầu đã nỗ lực để giấy phép của họ mang tính rõ ràng cao và có giá trị chung, bởi xét cho cùng, mỗi một người dùng đều nên hiểu về giấy phép và nếu cần có thể tự mình sử dụng chúng cho những phát triển của riêng họ.

Tuy nhiên, không khó để thấy được lý tưởng đơn giản này rất khó có thể đạt được vì có quá nhiều yêu cầu cụ thể và những hiểu biết pháp lý không phải lúc nào cũng tương thích lẫn nhau trên phạm vi quốc tế. Một ví dụ: Luật bản quyền của Đức và Mỹ về cơ bản là khác nhau. Theo luật của Đức, chỉ có một *người* là *tác giả* (trong tiếng Đức là *Urheber*), và tác phẩm của người đó là *tài sản trí tuệ* của họ. Mặc dù tác giả có thể cấp quyền sử dụng tác phẩm của mình nhưng họ lại không thể chuyển nhượng hoặc từ bỏ quyền tác giả của mình. Việc từ bỏ tác quyền là một điều khá xa lạ đối với luật pháp Hoa Kỳ. Ở Mỹ, một tác giả (tuy nhiên cũng có thể là một công ty hoặc một tổ chức) chỉ có quyền khai thác và có thể chuyển giao một phần hoặc toàn bộ quyền này, từ đó hoàn toàn tách mình ra khỏi tác phẩm của chính mình. Giấy phép có giá trị quốc tế phải được diễn giải trên cơ sở tôn trọng các quy định pháp luật khác nhau.

Hệ quả là có rất nhiều các loại giấy phép FOSS, đôi khi còn có nội dung hoàn toàn khác nhau. Tệ hơn nữa là còn có nhiều phiên bản khác nhau của cùng một loại giấy phép, hoặc hỗn hợp các giấy phép (trong một dự án hoặc thậm chí khi kết nối nhiều dự án), từ đó có thể gây ra nhầm lẫn hoặc thậm chí là tranh chấp pháp lý.

Cả những đại diện của phần mềm tự do và những người ủng hộ phong trào mã nguồn mở với định hướng kinh tế rất rõ ràng đã tạo ra các tổ chức của riêng họ mà ngày nay chịu trách nhiệm quyết

định về việc xây dựng các giấy phép phần mềm theo các nguyên tắc của họ cũng như hỗ trợ các thành viên của mình trong việc thực thi chúng.

Copyleft

Tổ chức Phần mềm Tự do (FSF) được nhắc đến ở trên đã xây dựng *Giấy phép Công cộng Chung GNU* (GPL) - một trong những giấy phép quan trọng nhất cho phần mềm tự do được sử dụng bởi nhiều dự án (ví dụ như Dự án Nhân Linux). Ngoài ra, tổ chức này cũng đã phát hành các giấy phép với các tùy chỉnh theo từng trường hợp cụ thể (chẳng hạn như *Giấy phép Công cộng Chung GNU Hạn chế* (LGPL) nhằm quản lý sự kết hợp của phần mềm tự do với các sửa đổi được thực hiện để mã hoá các phần mã nguồn của các sửa đổi không cần phải được phát hành công khai); *Giấy phép Công cộng Chung Affero GNU* (AGPL) bao gồm việc bán quyền truy cập vào phần mềm được lưu trữ, hoặc *Giấy phép Tài liệu Tự do GNU* (FDL) mở rộng các nguyên tắc tự do đối với tài liệu phần mềm. Ngoài ra, FSF cũng đưa ra các khuyến nghị ủng hộ hoặc chống lại giấy phép của bên thứ ba và các dự án liên kết như GPL-Violations.org có nhiệm vụ điều tra các vi phạm đáng ngờ đối với giấy phép tự do.

FSF gọi nguyên tắc mà theo đó giấy phép tự do cũng áp dụng cho các biến thể sửa đổi của phần mềm là *copyleft* — trái ngược với nguyên tắc hạn chế bản quyền (copyright) mà FSF bác bỏ. Do đó, ý tưởng ở đây là chuyển các nguyên tắc tự do của giấy phép phần mềm một cách không hạn chế nhất có thể sang các biến thể tương lai của chính phần mềm đó để ngăn chặn các hạn chế sau này.

Chuyện nghe có vẻ rõ ràng và đơn giản như vậy lại dẫn đến nhiều vấn đề phức tạp đáng kể trong thực tế. Đó là lý do tại sao các nhà phê bình thường gọi nguyên tắc copyleft là “nguyên tắc lan truyền” bởi nó sẽ được truyền tiếp sang các phiên bản sau đó.

Ví dụ, từ những gì đã được đề cập, ta có thể suy ra rằng hai thành phần phần mềm được cấp phép theo các giấy phép copyleft khác nhau có thể sẽ không kết hợp được với nhau vì không thể chuyển giao cả hai giấy phép cùng một lúc sang một sản phẩm tiếp theo. Điều này thậm chí cũng áp dụng với các phiên bản khác nhau của cùng một giấy phép!

Vì lý do này, các giấy phép hoặc phiên bản giấy phép mới hơn thường không nắm quyền copyleft một cách chặt chẽ nữa. Trên phương diện này, *Giấy phép Công cộng Hạn chế GNU* (LGPL) đã đề cập ở trên chính là một sự nhượng bộ để có thể kết nối phần mềm tự do với các thành phần “không tự do”; quy cách này thường được thực hiện với tên gọi là *thư viện*. Các thư viện này chứa các chương trình con hoặc thường trình được sử dụng bởi nhiều chương trình khác. Điều này dẫn đến một tình huống khá phổ biến là phần mềm độc quyền sử dụng một chương trình con như vậy từ một thư viện tự do.

Một cách khác để tránh xung đột giấy phép là *cấp phép kép*, trong đó, một phần mềm sẽ được cấp phép theo các giấy phép khác nhau (ví dụ: một giấy phép miễn phí và một giấy phép độc quyền).

Trường hợp sử dụng điển hình là phiên bản miễn phí của phần mềm chỉ có thể được sử dụng khi tôn trọng các hạn chế về quyền copyleft và các tùy chọn thay thế để có được phần mềm theo một giấy phép khác giúp giải phóng người dùng khỏi một số hạn chế nhất định, song song đối lại một khoản phí có thể được sử dụng để tài trợ cho việc phát triển phần mềm.

Do đó, rõ ràng là việc lựa chọn giấy phép cho các dự án phần mềm nên được thực hiện một cách hết sức thận trọng vì sự hợp tác với các dự án khác, khả năng kết hợp với các thành phần khác và cả thiết kế của sản phẩm tương lai của những phần mềm đó đều phụ thuộc vào điều này. Copyleft đã đặt ra cho các nhà phát triển những thách thức đặc biệt về khía cạnh này.

Định nghĩa về Mã nguồn mở và Giấy phép Linh hoạt

Về phía mã nguồn mở, chúng ta có *Tổ chức Sáng kiến Mã nguồn mở* (OSI) được thành lập vào năm 1998 bởi Eric S. Raymond và Bruce Perens, chủ yếu tập trung đến các vấn đề cấp phép. OSI cũng đã phát triển một quy trình đã được tiêu chuẩn hóa để kiểm tra giấy phép phần mềm tuân thủ *Định nghĩa Mã nguồn mở* của họ. Hơn 80 giấy phép mã nguồn mở được công nhận hiện có thể được tìm thấy trên trang web của OSI.

Tại đây, họ cũng liệt kê các giấy phép đạt tiêu chuẩn “OSI” (OSI-approved) mâu thuẫn rõ ràng với nguyên tắc copyleft, đặc biệt là nhóm *Giấy phép BSD*. *Bản phân phối Phần mềm Berkeley* (BSD) là một biến thể của hệ điều hành Unix ban đầu được phát triển tại Đại học Berkeley và sau này đã tạo ra các dự án tự do như *NetBSD*, *FreeBSD* và *OpenBSD*. Giấy phép nền tảng của các dự án này thường được gọi là *giấy phép linh hoạt*. Ngược lại với giấy phép copyleft, chúng không có mục đích thiết lập các điều khoản sử dụng cho các biến thể đã được sửa đổi. Thay vào đó, quyền tự do tối đa sẽ giúp phần mềm được phân phối rộng rãi nhất có thể bằng cách để những người chỉnh sửa phần mềm tự quyết định xử lý các chỉnh sửa như thế nào — chẳng hạn như liệu họ có muốn phát hành chúng hay coi chúng là một tài nguyên đóng và phân phối chúng dưới hình thức thương mại hoá.

Giấy phép BSD 2 Điều khoản, còn được gọi là *Giấy phép BSD đơn giản hóa* hoặc *Giấy phép BSD tự do*, là minh chứng cho mức độ đơn giản hoá của các giấy phép linh hoạt. Ngoài điều khoản trách nhiệm pháp lý được tiêu chuẩn hóa giúp bảo vệ nhà phát triển khỏi các khiếu nại trách nhiệm pháp lý phát sinh từ thiệt hại do phần mềm gây ra, giấy phép này chỉ bao gồm hai quy tắc sau:

Việc phân phối lại và sử dụng ở dạng nguồn và dạng nhị phân có hoặc không có sửa đổi được cho phép khi đáp ứng các điều kiện sau:

1. Việc phân phối lại mã nguồn phải giữ lại thông báo bản quyền ở trên, danh sách các điều kiện này và tuyên bố từ chối trách nhiệm sau đây.
2. Việc phân phối lại ở dạng nhị phân phải sao chép lại thông báo bản quyền ở trên, danh sách các điều kiện này và tuyên bố từ chối trách nhiệm sau đây trong tài liệu và/hoặc các thành phần khác được cung cấp cùng với việc phân phối.

Creative Commons

Ý tưởng phát triển thành công của FLOSS cũng như những tiến bộ công nghệ liên quan đã dẫn đến các nỗ lực chuyển giao nguyên tắc mã nguồn mở sang các lĩnh vực phi kỹ thuật khác. Việc chuẩn bị và cung cấp kiến thức cũng như sự hợp tác sáng tạo trong việc giải quyết các nhiệm vụ phức tạp hiện được coi là minh chứng của nguyên tắc mã nguồn mở mở rộng và chú trọng nội dung.

Điều này dẫn đến nhu cầu tạo ra các nền tảng đáng tin cậy cho cả các lĩnh vực này để theo đó các kết quả công việc có thể được chia sẻ và xử lý. Do các giấy phép phần mềm có sẵn hầu như đều không phù hợp với việc này nên đã có nhiều nỗ lực trong việc chuyển đổi các yêu cầu đặc thù này thành các giấy phép tiện dụng tương tự cho từ công trình khoa học tới các tác phẩm nghệ thuật số hóa “theo tinh thần của mã nguồn mở”.

Cho đến nay, sáng kiến quan trọng nhất thuộc phạm trù này là *Creative Commons* (CC) với các trọng điểm được tóm tắt như sau:

Creative Commons là một tổ chức phi lợi nhuận toàn cầu cho phép chia sẻ và sử dụng lại sự sáng tạo và kiến thức thông qua việc cung cấp các công cụ pháp lý miễn phí.

— <https://creativecommons.org/faq/#what-is-creative-commons-and-what-do-you-do>

Với Creative Commons, trọng tâm của việc chuyển nhượng quyền quay ngược từ nhà phân phối sang tác giả. Ví dụ: Trong xuất bản truyền thống, một tác giả thường chuyển giao tất cả các quyền xuất bản (in ấn, dịch thuật, v.v.) cho một nhà xuất bản; nhà xuất bản này sẽ đảm bảo việc phân phối tác phẩm một cách tốt nhất có thể. Các kênh phân phối Internet với những thay đổi đáng kể giờ đây đã đặt tác giả vào vị trí tự mình thực hiện các quyền xuất bản này và tự quyết định tác phẩm của mình có thể được sử dụng như thế nào. Creative Commons đã tạo cơ hội để xác định những vấn đề này một cách đơn giản và đáng tin cậy về mặt pháp lý và thậm chí còn nhiều hơn thế: các tác giả được khuyến khích cung cấp tác phẩm của họ như một đóng góp cho quá trình trao đổi và hợp tác chung. Không giống như tác quyền truyền thống là cung cấp cho tác giả tất cả các quyền mà họ có thể chuyển giao cho người khác khi cần, cách tiếp cận của Creative Commons là hoàn toàn ngược lại: tác giả cung cấp tác phẩm của mình cho cộng đồng, nhưng họ cũng có thể chọn một tập hợp các tính năng cần được cân nhắc khi sử dụng tác phẩm của họ — các tác giả càng chọn nhiều tính năng thì giấy phép càng trở nên hạn chế.

Và do đó, nguyên tắc “Chọn giấy phép” của CC sẽ yêu cầu tác giả từng bước xác định các thuộc tính riêng lẻ và tạo các giấy phép phù hợp mà sau này họ có thể gán cho tác phẩm dưới dạng văn bản và biểu tượng.

Để dễ hiểu hơn, sau đây là tổng quan về sáu loại kết hợp và giấy phép CC có thể cung cấp:

CC BY (“Attribution”)

Giấy phép tự do cho phép bất kỳ ai cũng có thể chỉnh sửa và phân phối tác phẩm miễn là họ có nêu tên tác giả.

CC BY-SA (“Attribution-ShareAlike”)

Giống như CC BY, ngoại trừ việc tác phẩm đã sửa đổi chỉ có thể được phân phối theo cùng một giấy phép. Nguyên tắc này nhắc ta nhớ về copyleft vì giấy phép này cũng là một loại giấy phép “kế thừa”.

CC BY-ND (“Attribution-NoDerivatives”)

Giống như CC BY, ngoại trừ việc tác phẩm chỉ có thể được chuyển giao khi không trải qua chỉnh sửa.

CC BY-NC (“Attribution-NonCommercial”)

Tác phẩm có thể được chỉnh sửa và phân phối bằng cách nêu tên tác giả nhưng chỉ trong các điều kiện phi thương mại.

CC BY-NC-SA (“Attribution-NonCommercial-ShareAlike”)

Giống như BY-NC, ngoại trừ việc tác phẩm chỉ có thể được chia sẻ với cùng các điều kiện (ví như cùng một giấy phép giống với copyleft).

CC BY-NC-ND (“Attribution-NonCommercial-NoDerivatives”)

Giấy phép hạn chế nhất: việc phân phối được cho phép với sự ghi công của tác giả nhưng chỉ được phân phối khi không trải qua chỉnh sửa và trong các điều kiện phi thương mại.

Mô hình kinh doanh trong Mã nguồn mở

Khi nhìn lại, chiến thắng của FLOSS đóng vai trò như một phong trào gốc rễ của những người theo chủ nghĩa công nghệ lý tưởng không phụ thuộc vào những ràng buộc kinh tế, tiền tệ và đưa các tác phẩm của họ vào phục vụ công chúng. Đồng thời, các công ty trị giá hàng tỷ đô la đã được tạo ra trong môi trường của FLOSS; một ví dụ điển hình chính là công ty *Red Hat* của Mỹ thành lập năm 1993 với doanh thu hàng năm lên tới hơn 3 tỷ USD (2018) và đã được gã khổng lồ CNTT IBM tiếp quản vào năm 2018.

Vì vậy, trên lập trường của những nhà sáng tạo và sáng lập, chúng ta phải xem xét sự căng thẳng giữa việc phân phối phần mềm chất lượng cao miễn phí và gần như miễn phí với các mô hình kinh doanh vì một lý do rất rõ ràng: vô số các nhà phát triển phần mềm tự do có trình độ cao cũng cần phải kiếm được tiền, và do đó, môi trường FLOSS phi thương mại thuần túy ban đầu phải phát triển các mô hình kinh doanh bền vững để bảo tồn vũ trụ của chính nó.

Một cách tiếp cận phổ biến, đặc biệt là đối với các dự án lớn hơn trong giai đoạn ban đầu, được gọi là *huy động vốn từ cộng đồng*, tức là thu tiền quyền góp thông qua một nền tảng như *Kickstarter*. Đối lại, các nhà tài trợ sẽ nhận được một phần thưởng được định trước từ các nhà phát triển trong trường hợp dự án thành công, tức là nếu đạt được các mục tiêu đã xác định trước đó, có thể là quyền truy cập không giới hạn vào sản phẩm hoặc các tính năng đặc biệt.

Một cách tiếp cận khác là *cấp phép kép*: phần mềm tự do sẽ được cung cấp song song theo một giấy phép hạn chế hơn hoặc thậm chí là độc quyền, đồng thời đảm bảo cho khách hàng các dịch vụ mở rộng hơn (thời gian phản hồi trong trường hợp có lỗi, các cập nhật, phiên bản cho các nền tảng nhất định, v.v.). Một ví dụ chính là *ownCloud* đang được phát triển bởi GPL và cung cấp cho khách hàng doanh nghiệp “Phiên bản doanh nghiệp” theo giấy phép độc quyền.

Chúng ta cũng hãy lấy *ownCloud* làm ví dụ về một mô hình kinh doanh FLOSS phổ biến khác: các dịch vụ chuyên nghiệp. Nhiều công ty có thiếu sót về kiến thức kỹ thuật nội bộ cần thiết để có thể thiết lập cũng như vận hành các phần mềm quan trọng và phức tạp một cách đáng tin cậy và trên hết là an toàn. Đó là lý do tại sao họ mua các dịch vụ chuyên nghiệp như tư vấn, bảo trì hoặc trợ giúp trực tiếp từ các nhà sản xuất. Các vấn đề về trách nhiệm pháp lý cũng đóng một vai trò nhất định trong quyết định này bởi các công ty cũng có thể chuyển giao các rủi ro hoạt động tới nhà sản xuất.

Nếu một phần mềm có thể trở nên thành công và phổ biến trong lĩnh vực của nó thì đó là do khi khách hàng sử dụng các phần mềm này, có khả năng kiếm tiền ngoại vi (như buôn bán hoặc sử dụng các chứng chỉ) từ đó và công nhận địa vị cụ thể của chúng. Chẳng hạn như nền tảng học tập *Moodle* cung cấp chứng nhận cho các giảng viên và những người lưu lại kiến thức của họ cho các khách hàng tiềm năng. Đây chỉ là một ví dụ trong số vô vàn các ví dụ khác.

Phần mềm dưới dạng Dịch vụ (SaaS) là một mô hình kinh doanh khác đặc biệt dành cho các công nghệ dựa trên trang web. Tại đây, nhà cung cấp đám mây sẽ chạy các phần mềm như Phần mềm Quản lý Quan hệ Khách hàng (CRM) hoặc Hệ thống quản lý nội dung (CMS) trên máy chủ và cấp cho khách hàng của họ quyền truy cập vào ứng dụng đã cài đặt. Điều này giúp cho khách hàng không cần bận tâm tới quá trình cài đặt và bảo trì phần mềm. Đối lại, khách hàng sẽ trả tiền cho việc sử dụng phần mềm dựa trên các thông số (ví dụ như số lượng người dùng). Tính khả dụng và bảo mật đóng một vai trò quan trọng bởi chúng chính là các yếu tố kinh doanh trọng yếu.

Cuối cùng nhưng không kém phần quan trọng là mô hình phát triển các tiện ích mở rộng dành riêng cho từng khách hàng lên thành các phần mềm tự do theo yêu cầu; mô hình này đặc biệt phổ biến trong các dự án nhỏ. Sau đó, khách hàng thường tự quyết định tiếp việc sử dụng và phân phối các tiện ích mở rộng này, tức là họ cũng có thể phát hành chúng hoặc khoá chúng lại như một phần của mô hình kinh doanh của riêng mình.

Một điều đáng lẽ phải rõ ràng hơn là mặc dù các phần mềm tự do thường có sẵn miễn phí nhưng

nhiều mô hình kinh doanh đã được tạo ra trong môi trường của chúng. Những mô hình này liên tục được sửa đổi và mở rộng bởi vô số các tác giả tự do và các công ty trên toàn thế giới dưới những hình thức rất sáng tạo, và toàn bộ những yếu tố này chính là sự bảo đảm cho sự tồn tại của toàn bộ phong trào FLOSS.

Bài tập Hướng dẫn

1. Tóm lại, “bốn quyền tự do” theo định nghĩa của Richard Stallman và Tổ chức Phần mềm Tự do là gì?

quyền tự do số 0	
quyền tự do số 1	
quyền tự do số 2	
quyền tự do số 3	

2. FLOSS là từ viết tắt đại diện cho cái gì?

3. Bạn đã phát triển phần mềm tự do và muốn đảm bảo rằng bản thân phần mềm đó cũng như tất cả các hoạt động trong tương lai dựa trên nó sẽ vẫn tiếp tục được tự do. Bạn sẽ chọn giấy phép nào?

CC BY	
GPL Phiên bản 3	
Giấy phép BSD 2 điều khoản	
LGPL	

4. Bạn gọi giấy phép nào sau đây là giấy phép linh hoạt, giấy phép nào là copyleft?

Giấy phép BSD đơn giản hóa	
GPL Phiên bản 3	
CC BY	
CC BY-SA	

5. Bạn đã viết một ứng dụng web và xuất bản nó theo giấy phép tự do. Làm thế nào để bạn có thể kiếm tiền từ sản phẩm này? Hãy nêu ra ba khả năng.

Bài tập Mở rộng

1. Các ứng dụng sau có sẵn theo giấy phép nào (bao gồm cả phiên bản)?

Apache HTTP Server	
MySQL Community Server	
Wikipedia articles	
Mozilla Firefox	
GIMP	

2. Bạn muốn phát hành phần mềm của mình theo GNU GPL Phiên bản 3. Bạn nên làm theo những bước nào?

3. Bạn đã viết phần mềm độc quyền và muốn kết hợp nó với phần mềm tự do theo GPL phiên bản 3. Bạn có được phép làm điều này không hay sẽ phải cân nhắc điều gì?

4. Tại sao Tổ chức Phần mềm Tự do lại phát hành *Giấy phép Công cộng Chung Affero GNU* (GNU AGPL) như một phần bổ sung cho GNU GPL?

5. Hãy kể tên ba ví dụ về phần mềm tự do được cung cấp dưới dạng “Phiên bản doanh nghiệp”, tức phiên bản có tính phí.

Tóm tắt

Trong bài học này chúng ta đã học về:

- Điểm giống và khác nhau giữa Phần mềm tự do và Phần mềm mã nguồn mở (FLOSS)
- Giấy phép FLOSS, tầm quan trọng và những vấn đề xoay quanh chúng
- Copyleft và giấy phép linh hoạt
- Mô hình kinh doanh FLOSS

Đáp án Bài tập Hướng dẫn

1. Tóm lại, “bốn quyền tự do” theo định nghĩa của Richard Stallman và Tổ chức Phần mềm Tự do là gì?

quyền tự do số 0	chạy phần mềm
quyền tự do số 1	nghiên cứu và sửa đổi phần mềm (mã nguồn)
quyền tự do số 2	phân phối phần mềm
quyền tự do số 3	phân phối phần mềm đã qua sửa đổi

2. FLOSS là từ viết tắt đại diện cho cái gì?

Free/Libre Open Source Software - Phần mềm Mã nguồn Mở Tự do.

3. Bạn đã phát triển phần mềm tự do và muốn đảm bảo rằng bản thân phần mềm đó cũng như tất cả các hoạt động trong tương lai dựa trên nó sẽ vẫn tiếp tục được miễn phí. Bạn sẽ chọn giấy phép nào?

CC BY	
GPL Phiên bản 3	X
Giấy phép BSD 2 điều khoản	
LGPL	

4. Bạn gọi giấy phép nào sau đây là giấy phép linh hoạt, giấy phép nào là copyleft?

Giấy phép BSD đơn giản hóa	linh hoạt
GPL Phiên bản 3	copyleft
CC BY	linh hoạt
CC BY-SA	copyleft

5. Bạn đã viết một ứng dụng web và xuất bản nó theo giấy phép tự do. Làm thế nào để bạn có thể kiếm tiền từ sản phẩm này? Hãy nêu ra ba khả năng.

- Cấp phép kép, ví dụ: bằng cách cung cấp “Phiên bản doanh nghiệp” có tính phí.
- Cung cấp lưu trữ, dịch vụ và hỗ trợ.
- Phát triển tiện ích mở rộng độc quyền cho khách hàng.

Đáp án Bài tập Mở rộng

1. Các ứng dụng sau có sẵn theo giấy phép nào (bao gồm cả phiên bản)?

Apache HTTP Server	Apache License 2.0
MySQL Community Server	GPL 2
Wikipedia articles (English)	Creative Commons Attribution Share-Alike license (CC-BY-SA)
Mozilla Firefox	Mozilla Public License 2.0
GIMP	GPL 3

2. Bạn muốn phát hành phần mềm của mình theo GNU GPL Phiên bản 3. Bạn nên làm theo những bước nào?

- Nếu cần, hãy bảo vệ bản thân trước người sử dụng dịch vụ, có thể là bằng cách từ bỏ bản quyền để bạn có thể chỉ định giấy phép.
- Thêm một thông báo bản quyền cho mỗi tệp.
- Thêm tệp có tên COPYING với toàn bộ nội dung giấy phép vào phần mềm của bạn.
- Thêm tham chiếu đến giấy phép trong mỗi tệp.

3. Bạn đã viết phần mềm độc quyền và muốn kết hợp nó với phần mềm tự do theo GPL phiên bản 3. Bạn có được phép làm điều này không hay sẽ phải cân nhắc điều gì?

Mục Các câu hỏi thường gặp của Tổ chức Phần mềm Tự do có cung cấp thông tin như sau: Sự kết hợp là khả thi với điều kiện phần mềm độc quyền và phần mềm miễn phí của bạn vẫn tách biệt với nhau. Tuy nhiên, bạn phải chắc chắn rằng sự tách biệt này vẫn được đảm bảo về mặt kỹ thuật và người dùng có thể nhận biết được. Nếu tích hợp phần mềm miễn phí bằng cách biến nó trở thành một phần của sản phẩm của bạn thì bạn cũng phải xuất bản sản phẩm với giấy phép GPL theo nguyên tắc copyleft.

4. Tại sao Tổ chức Phần mềm Tự do lại phát hành *Giấy phép Công cộng Chung Affero GNU* (GNU AGPL) như một phần bổ sung cho GNU GPL?

GNU AGPL thu hẹp khoảng cách về giấy phép đặc biệt sẽ phát sinh với phần mềm tự do được lưu trữ trên máy chủ: Nếu nhà phát triển thực hiện các thay đổi đối với phần mềm, họ không bắt buộc phải công khai quyền truy cập cho những thay đổi này theo GPL bởi họ đã cho phép truy cập vào chương trình chứ không hề “phân phối lại” dựa trên chương trình theo định nghĩa GPL. Còn GNU AGPL lại quy định rằng phần mềm phải được cung cấp để tải xuống cùng với tất cả các thay đổi đã được thực hiện.

5. Hãy kể tên ba ví dụ về phần mềm tự do được cung cấp dưới dạng “Phiên bản doanh nghiệp”, tức phiên bản có tính phí.

MySQL, Zammad, Nextcloud



**Linux
Professional
Institute**

1.4 Kỹ năng CNTT và Làm việc trên Linux

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 1.4](#)

Khối lượng

2

Các lĩnh vực kiến thức chính

- Kỹ năng về Máy tính để bàn
- Vào dòng lệnh
- Ngành công nghiệp sử dụng Linux, điện toán đám mây và ảo hóa

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

- Sử dụng trình duyệt, quan ngại về quyền riêng tư, tùy chọn cấu hình, tìm kiếm trên web và lưu nội dung
- Thiết bị đầu cuối và bảng điều khiển
- Các vấn đề về mật khẩu
- Các vấn đề và công cụ về quyền riêng tư
- Sử dụng các ứng dụng mã nguồn mở phổ biến trong các bài thuyết trình và dự án



Linux
Professional
Institute

1.4 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	1 Cộng đồng Linux và Sự nghiệp trong Mã nguồn mở
Mục tiêu:	1.4 Kỹ năng CNTT-TT và Làm việc trên Linux
Bài:	1 trên 1

Giới thiệu

Đã có một thời gian việc làm việc với Linux trên máy tính để bàn được coi là khó khăn bởi hệ thống của nó thiếu nhiều ứng dụng và công cụ cấu hình cao cấp dành cho máy tính để bàn so với các hệ điều hành khác. Một số lý do để giải thích cho việc này là bởi Linux khi đó còn "non" hơn rất nhiều so với nhiều hệ điều hành khác. Do đó, sẽ dễ dàng hơn khi Linux bắt đầu bằng cách phát triển các ứng dụng dòng lệnh thiết yếu và để dành các công cụ đồ họa phức tạp hơn cho sau này. Ban đầu, vì Linux vốn nhắm mục tiêu đến những người dùng cao cấp hơn nên đó không phải là vấn đề. Nhưng những ngày đó đã qua từ lâu. Ngày nay, môi trường máy tính để bàn Linux đã trưởng thành hơn nhiều và không còn các nhu cầu về các tính năng và tính dễ sử dụng nữa. Tuy nhiên, dòng lệnh vẫn được coi là một công cụ mạnh mẽ được những người dùng nâng cao sử dụng hàng ngày. Trong bài học này, chúng ta sẽ xem xét một số kỹ năng cơ bản về máy tính để bàn mà bạn cần để chọn ra được công cụ tốt nhất cho công việc của bạn, bao gồm cả việc truy cập vào dòng lệnh.

Giao diện Người dùng Linux

Khi sử dụng hệ thống Linux, bạn có thể tương tác với dòng lệnh hoặc với giao diện người dùng đồ họa. Cả hai cách đều cấp cho bạn quyền truy cập vào nhiều ứng dụng hỗ trợ thực hiện hầu hết mọi tác vụ với máy tính. Mặc dù mục tiêu 1.2 đã giới thiệu cho bạn một loạt các ứng dụng thường được sử dụng, nhưng chúng ta sẽ bắt đầu bài học này bằng cách xem xét kỹ hơn về môi trường máy tính để bàn, cách truy cập cửa sổ dòng lệnh và các công cụ được sử dụng để trình chiếu và quản lý dự án.

Môi trường Máy tính để bàn

Linux có cách tiếp cận mô-đun mà trong đó các phần khác nhau của hệ thống được phát triển bởi các dự án và nhà phát triển khác nhau, mỗi dự án đáp ứng một nhu cầu hoặc một mục tiêu cụ thể. Do đó, có khá nhiều tùy chọn cho môi trường máy tính để bàn; song song với trình quản lý gói thì môi trường máy tính để bàn mặc định là một trong những điểm khác biệt chính giữa nhiều những bản phân phối hiện có. Không giống như các hệ điều hành độc quyền như Windows và macOS - nơi mà người dùng bị giới hạn trong môi trường máy tính đi kèm với hệ điều hành của họ - với Linux, bạn có thể cài đặt nhiều môi trường và chọn một môi trường phù hợp nhất với bản thân và nhu cầu của mình.

Về cơ bản, có hai môi trường máy tính để bàn chính trong vũ trụ Linux: *Gnome* và *KDE*. Cả hai đều rất hoàn chỉnh và có một cộng đồng lớn đứng đằng sau cũng như cùng hướng đến một mục đích dù có các cách tiếp cận hơi khác nhau. Tóm lại, Gnome cố gắng tuân theo nguyên tắc KISS (“Keep It Simple, Stupid” - càng đơn giản càng tốt) với các ứng dụng rất hợp lý và gọn gàng. KDE thì lại có một quan điểm khác cùng với nhiều lựa chọn ứng dụng hơn và cung cấp cho người dùng cơ hội thay đổi mọi cài đặt cấu hình trong môi trường.

Trong khi các ứng dụng Gnome dựa trên bộ công cụ GTK (được viết bằng ngôn ngữ C) thì các ứng dụng KDE sử dụng thư viện Qt (được viết bằng C++). Một trong những khía cạnh thực tế nhất của việc viết các ứng dụng với cùng một bộ công cụ đồ họa là các ứng dụng sẽ có xu hướng có giao diện tương tự nhau, mang lại cho người dùng cảm giác thống nhất trong trải nghiệm của họ. Một đặc điểm quan trọng khác là việc có cùng một thư viện đồ họa được chia sẻ cho nhiều ứng dụng khác nhau cùng được sử dụng thường xuyên có thể tiết kiệm được dung lượng bộ nhớ, đồng thời sẽ tăng tốc thời gian tải thư viện sau khi đã tải lần đầu.

Truy cập vào Dòng lệnh

Đối với chúng ta, một trong những ứng dụng quan trọng nhất là trình giả lập cửa sổ dòng lệnh đồ họa. Chúng được gọi là trình giả lập cửa sổ dòng lệnh vì chúng thực sự mô phỏng cửa sổ dòng lệnh nối tiếp kiểu cũ (thường là máy Teletype) trong môi trường đồ họa. Trên thực tế, các thiết bị này là các máy khách từng được kết nối với một máy từ xa nơi các thuật toán thực sự được diễn ra.

Những chiếc máy đó thực chất là những chiếc máy tính đơn giản không có đồ họa được sử dụng trên các phiên bản đầu tiên của Unix.

Trong Gnome, một ứng dụng như vậy được gọi là *Cửa sổ dòng lệnh Gnome*; trong khi đó, ở KDE, nó có thể được tìm thấy dưới cái tên *Konsole*. Ngoài ra, ta cũng có nhiều lựa chọn khác có sẵn khác như *Xterm*. Các ứng dụng đó là một cách để ta có được quyền truy cập vào môi trường dòng lệnh để có thể tương tác với vỏ (shell).

Vì vậy, bạn nên xem menu ứng dụng của bản phân phối bạn đã chọn cho ứng dụng đầu cuối. Dù mọi ứng dụng đều có những điểm khác nhau, chúng đều sẽ cung cấp cho bạn những gì cần thiết để có thể tự tin sử dụng dòng lệnh.

Một cách khác để truy cập vào cửa sổ dòng lệnh là sử dụng TTY ảo. Bạn có thể truy cập chúng bằng cách nhấn tổ hợp `Ctrl + Alt + F#`. Ví dụ: coi `F#` là một trong các phím chức năng từ 1 đến 7. Một số tổ hợp ban đầu có thể đang chạy trình quản lý phiên hoặc môi trường đồ họa của bạn. Những tổ hợp còn lại sẽ hiển thị một dấu nhắc lệnh yêu cầu tên đăng nhập của bạn như dưới đây:

```
Ubuntu 18.10 arrelia tty3
arrelia login:
```

`arrelia` trong trường hợp này là tên máy chủ của máy và `tty3` là cửa sổ dòng lệnh khả dụng sau khi sử dụng tổ hợp phím ở trên, cộng với phím `F3` như trong tổ hợp `Ctrl + Alt + F3`.

Sau khi cung cấp thông tin đăng nhập và mật khẩu, bạn sẽ vào được một cửa sổ dòng lệnh, nhưng ở đây sẽ không có môi trường đồ họa; vì vậy, bạn sẽ không thể sử dụng chuột hoặc chạy các ứng dụng đồ họa mà không khởi động phiên X hoặc Wayland trước. Nhưng vấn đề này nằm ngoài phạm vi của bài học này.

Trình chiếu và Dự án

Công cụ quan trọng nhất để trình chiếu trên Linux là *LibreOffice Impress*. Nó là một phần của bộ ứng dụng văn phòng mã nguồn mở có tên *LibreOffice*. Hãy coi LibreOffice như một sự thay thế tương đương của mã nguồn mở cho *Microsoft Office*. Nó thậm chí có thể mở và lưu các tệp PPT và PPTX có nguồn gốc từ *Powerpoint*. Nhưng bất chấp điều này, định dạng ODP Impress gốc vẫn được khuyến khích. ODP là một phần của *Open Document Format*, tức tiêu chuẩn quốc tế cho loại tệp này. Điều này đặc biệt quan trọng nếu bạn muốn giữ cho tài liệu của mình có thể truy cập được trong vòng nhiều năm và giảm lo ngại về các sự cố tương thích. Bởi vì chúng là một tiêu chuẩn mở, bất kỳ ai cũng có thể triển khai định dạng này mà không phải trả bất kỳ khoản tiền nào cho bản quyền hoặc giấy phép. Điều này cũng giúp bạn thoải mái dùng thử các phần mềm thuyết trình khác mới hơn mà bạn có thể thích hơn với các tệp của mình, vì rất có thể chúng sẽ tương thích với những phần mềm đó.

Nhưng nếu bạn thích viết mã hơn là giao diện đồ họa thì có một số công cụ để bạn lựa chọn. *Beamer* là một định dạng của *LaTeX* và có thể tạo bản trình chiếu từ mã *LaTeX*. Bản thân *LaTeX* là một hệ thống sắp chữ được sử dụng chủ yếu để viết các tài liệu khoa học tại học viện, đặc biệt do khả năng xử lý các ký hiệu toán học phức tạp - điều mà các phần mềm khác gặp khó khăn khi xử lý. Nếu bạn đang ở trường đại học và cần giải các phương trình cũng như các vấn đề liên quan đến toán học khác, *Beamer* có thể giúp bạn tiết kiệm rất nhiều thời gian.

Một lựa chọn khác là *Reveal.js* - một gói NPM tuyệt vời (NPM là trình quản lý gói NodeJS mặc định) cho phép bạn tạo các bản trình chiếu đẹp mắt bằng cách sử dụng web. Vì vậy, nếu bạn có thể viết HTML và CSS, *Reveal.js* sẽ mang tới hầu hết các JavaScript cần thiết để tạo các bản trình chiếu đẹp mắt và mang tính tương tác cùng với khả năng thích ứng tốt trên mọi độ phân giải và kích thước màn hình. Cuối cùng, nếu bạn muốn thay thế *Microsoft Project*, bạn có thể thử *GanttProject* hoặc *ProjectLibre*. Cả hai đều rất giống với đối thủ độc quyền của chúng và tương thích với các tệp Dự án.

Công dụng của Linux trong Công nghiệp

Linux được sử dụng nhiều trong ngành công nghiệp phần mềm và Internet. Các trang web như [W3Techs](#) báo cáo rằng khoảng 68% máy chủ trang web trên Internet được cung cấp bởi Unix và phần lớn nhất trong số đó là Linux.

Việc được áp dụng rộng rãi này không chỉ bởi tính tự do của Linux (cả về nghĩa "tự do ngôn luận" lẫn "vào cổng tự do") mà còn vì tính ổn định, linh hoạt và hiệu suất của nó. Những đặc điểm này cho phép các nhà cung cấp dịch vụ của họ với chi phí thấp hơn và khả năng mở rộng tốt hơn. Một phần đáng kể của các hệ thống Linux ngày nay chạy trên đám mây, có thể là trên mô hình IaaS (Infrastructure as a service - Cơ sở hạ tầng dưới dạng dịch vụ), PaaS (Platform as a Service - Nền tảng dưới dạng dịch vụ) hoặc SaaS (Software as a Service - Phần mềm dưới dạng dịch vụ).

IaaS là một cách để chia sẻ tài nguyên của một máy chủ lớn thông qua việc cung cấp cho họ quyền truy cập vào các máy ảo mà trên thực tế chính là nhiều hệ điều hành đang chạy với tư cách là khách trên máy chủ thông qua một phần mềm quan trọng được gọi là *trình giám sát máy ảo*. Trình giám sát máy ảo chịu trách nhiệm giúp các hệ điều hành khách có thể chạy được bằng cách tách biệt và quản lý các tài nguyên có sẵn trên máy chủ cho những khách đó. Tác vụ này chính là *ảo hóa*. Trong mô hình IaaS, bạn chỉ phải trả tiền cho phần tài nguyên mà cơ sở hạ tầng của bạn sử dụng.

Linux có ba trình giám sát máy ảo mã nguồn mở nổi tiếng: *Xen*, *KVM* và *VirtualBox*. *Xen* có lẽ là lớn tuổi nhất trong số chúng. *KVM* vượt qua *Xen* với tư cách là trình giám sát máy ảo mã nguồn mở Linux nổi bật nhất. Nó được RedHat tài trợ phát triển và được họ cùng nhiều những nhà phát triển khác sử dụng, cả trong các dịch vụ đám mây công cộng lẫn các thiết lập đám mây riêng.

VirtualBox thuộc về Oracle kể từ khi họ mua lại Sun Microsystems và thường được người dùng cuối sử dụng vì tính dễ sử dụng và quản trị của nó.

Mặt khác, PaaS và SaaS được xây dựng dựa trên mô hình IaaS cả về mặt kỹ thuật lẫn khái niệm. Trong PaaS, thay vì máy ảo, người dùng có quyền truy cập vào một nền tảng nơi họ có thể triển khai và chạy ứng dụng của mình. Mục tiêu ở đây là giảm bớt gánh nặng xử lý các tác vụ quản trị hệ thống và cập nhật của hệ điều hành. [Heroku](#) là một ví dụ PaaS phổ biến mà trong đó mã chương trình có thể chạy mà không cần quan tâm đến các vật chứa và máy ảo nền tảng.

Cuối cùng, SaaS là mô hình mà bạn thường trả phí đăng ký chỉ để sử dụng một phần mềm mà không phải lo lắng về bất kỳ điều gì khác. *Dropbox* và *Salesforce* là hai ví dụ điển hình về SaaS. Hầu hết các dịch vụ này được truy cập thông qua trình duyệt web.

Một dự án như *OpenStack* là một tập hợp các phần mềm mã nguồn mở có thể sử dụng các trình giám sát máy ảo và các công cụ khác nhau để cung cấp môi trường đám mây IaaS hoàn chỉnh tại chỗ thông qua việc tận dụng sức mạnh của cụm máy tính trên trung tâm dữ liệu của riêng bạn. Tuy nhiên, việc thiết lập cơ sở hạ tầng như vậy không phải là một chuyện dễ dàng.

Các vấn đề về Quyền riêng tư khi sử dụng Internet

Ngày nay, trình duyệt web là một phần mềm cơ bản trên bất kỳ loại máy tính để bàn nào, nhưng một số người dùng vẫn thiếu kiến thức để có thể sử dụng nó một cách an toàn. Mặc dù ngày càng có nhiều dịch vụ được truy cập thông qua trình duyệt web, nhưng hầu hết các tác vụ được thực hiện thông qua trình duyệt đều được theo dõi và phân tích bởi nhiều bên khác nhau. Đảm bảo quyền truy cập vào các dịch vụ internet và ngăn chặn khả năng bị theo dõi là một khía cạnh quan trọng của việc sử dụng internet một cách an toàn.

Cookie theo dõi

Giả sử bạn đã truy cập vào một trang web thương mại điện tử, đã chọn sản phẩm bạn muốn và cho nó vào giỏ hàng. Nhưng vào giây cuối cùng, bạn lại suy nghĩ lại và xem xét kỹ hơn một chút xem liệu bạn có thực sự cần món đồ đó hay không. Sau một thời gian, bạn bắt đầu thấy các quảng cáo về chính sản phẩm đó đi theo bạn trên trang web. Khi nhấp vào quảng cáo, bạn sẽ ngay lập tức được đưa trở lại trang sản phẩm của cửa hàng đó. Không có gì lạ khi các sản phẩm bạn đã đặt trong giỏ hàng vẫn ở đó, chỉ chờ bạn thanh toán. Bạn đã bao giờ tự hỏi họ làm điều đó bằng cách nào chưa? Họ làm thế nào để hiển thị đúng quảng cáo tới bạn tại một trang web khác? Câu trả lời cho những câu hỏi này chính là *cookie theo dõi*. Cookie là các tệp nhỏ mà một trang web có thể lưu trên máy tính của bạn để lưu trữ và truy xuất một số loại thông tin có thể hữu ích cho việc điều hướng của bạn. Chúng đã được sử dụng trong nhiều năm và là một trong những cách lâu đời nhất để lưu trữ dữ liệu ở phía máy khách. Một ví dụ điển hình về việc sử dụng chúng là thẻ nhận dạng mua sắm độc nhất. Bằng cách đó, nếu bạn quay lại cùng một trang web sau vài ngày, cửa

hàng có thể nhớ cho bạn những sản phẩm bạn đã đặt vào giỏ hàng trong lần truy cập trước và giúp bạn tiết kiệm thời gian tìm lại chúng.

Thông thường thì điều này khá là vô hại, vì trang web chỉ đang cung cấp cho bạn một tính năng hữu ích và không chia sẻ bất kỳ dữ liệu nào với bên thứ ba. Nhưng còn những quảng cáo được hiển thị cho bạn khi bạn lướt trên các trang web khác thì sao? Đó là nơi các mạng quảng cáo xuất hiện. Mạng quảng cáo là các công ty một mặt cung cấp quảng cáo cho các trang web thương mại điện tử giống như trang trong ví dụ trên, mặt khác là kiếm tiền từ các trang web. Ví dụ: những người tạo nội dung như các blogger có thể cung cấp một số không gian cho các mạng quảng cáo đó trên blog của họ để đổi lấy tiền hoa hồng liên quan đến doanh số bán hàng do quảng cáo đó tạo ra.

Nhưng làm sao họ biết nên giới thiệu sản phẩm nào cho bạn? Họ thường làm điều này bằng cách lưu một cookie từ mạng quảng cáo tại thời điểm bạn truy cập hoặc tìm kiếm một sản phẩm nhất định trên trang web thương mại điện tử. Bằng cách đó, mạng có thể truy xuất thông tin trên cookie đó ở bất cứ nơi nào mạng có quảng cáo, tạo mối tương quan với các sản phẩm bạn quan tâm. Đây thường là một trong những cách phổ biến nhất để theo dõi ai đó qua Internet. Ví dụ đã được đưa ra ở trên sử dụng thương mại điện tử để làm cho mọi thứ trở nên hữu hình hơn, nhưng các nền tảng truyền thông xã hội cũng làm như vậy với các nút “Thích” hoặc “Chia sẻ” cũng như thông tin đăng nhập mạng xã hội của họ.

Một cách để bạn có thể loại bỏ điều này là không cho phép các trang web của bên thứ ba lưu trữ cookie trên trình duyệt của bạn. Bằng cách này, chỉ trang web bạn truy cập mới có thể lưu trữ cookie của họ. Nhưng hãy lưu ý rằng một số tính năng “hữu ích” có thể sẽ không hoạt động được nếu bạn làm điều này, bởi vì nhiều trang web ngày nay dựa vào các dịch vụ của bên thứ ba để hoạt động. Vì vậy, bạn có thể tìm kiếm trình quản lý cookie tại kho lưu trữ tiện ích bổ sung của trình duyệt để có quyền kiểm soát chi tiết những cookie nào đang được lưu trữ trên máy của bạn.

Không cho phép Theo dõi (DNT)

Một quan niệm sai lầm phổ biến khác có liên quan đến một cấu hình trình duyệt nhất định được gọi là DNT. Đó là từ viết tắt của “Do Not Track (Không cho phép Theo dõi)” và về cơ bản nó có thể được bật trên bất kỳ trình duyệt nào hiện có. Tương tự như chế độ riêng tư, không khó để tìm thấy những người tin rằng họ sẽ không bị theo dõi nếu họ bật cấu hình này. Thật không may, điều này không phải lúc nào cũng đúng. Hiện tại, DNT chỉ là một cách để bạn nói với các trang web bạn truy cập rằng bạn không muốn họ theo dõi bạn. Nhưng trên thực tế, họ mới là người quyết định xem họ có tôn trọng lựa chọn của bạn hay không. Nói cách khác, DNT là một cách để từ chối việc theo dõi của trang web, nhưng không có gì để đảm bảo cho lựa chọn này.

Về mặt kỹ thuật, điều này được thực hiện bằng cách gửi thêm một lá cờ trên tiêu đề của giao thức yêu cầu HTTP (DNT: 1) khi yêu cầu dữ liệu từ máy chủ web. Nếu bạn muốn biết thêm về chủ đề này, trang web <https://allaboutdnt.com> là một khởi đầu tốt.

Cửa sổ “Riêng tư”

Có thể bạn đã nhận ra rằng tiêu đề trên được đặt trong dấu trích dẫn kép. Đây là bởi những cửa sổ đó không hề riêng tư như hầu hết mọi người nghĩ. Tên có thể khác nhau nhưng chúng đều được gọi với những cái tên là tab “chế độ riêng tư (private mode)”, “giấu tên (incognito)” hoặc “ẩn danh (anonymous)”, tùy thuộc vào trình duyệt bạn đang sử dụng.

Trong Firefox, bạn có thể dễ dàng sử dụng nó bằng cách nhấn tổ hợp phím **Ctrl** + **Shift** + **P**. Trong Chrome, chỉ cần nhấn **Ctrl** + **Shift** + **N**. Những gì tác vụ này thực sự làm là mở ra một phiên làm việc hoàn toàn mới, phiên này thường không chia sẻ bất kỳ cấu hình hoặc dữ liệu nào từ hồ sơ tiêu chuẩn của bạn. Khi bạn đóng cửa sổ riêng tư, trình duyệt sẽ tự động xóa toàn bộ dữ liệu do phiên làm việc đó tạo ra, không để lại dấu vết trên máy tính sử dụng. Điều này có nghĩa là không có dữ liệu cá nhân nào (ví dụ như lịch sử, mật khẩu hoặc cookie) còn được lưu trữ trên máy tính đó.

Do vậy, nhiều người hiểu sai khái niệm này và tin rằng họ có thể truy cập ẩn danh trên Internet - điều này không hoàn toàn đúng. Một điều mà chế độ riêng tư hoặc ẩn danh thực sự thực hiện là tránh cookie theo dõi. Khi bạn truy cập một trang web, nó có thể lưu trữ một tệp nhỏ trên máy tính của bạn; tệp này có thể chứa ID có thể được sử dụng để theo dõi bạn. Trừ khi bạn định cấu hình trình duyệt của mình để không chấp nhận cookie của bên thứ ba, mạng quảng cáo hoặc các công ty khác có thể lưu trữ và truy xuất ID đó và thực sự theo dõi quá trình truy cập của bạn trên các trang web. Tuy nhiên, vì các cookie được lưu trữ trong phiên ở chế độ riêng tư sẽ bị xóa ngay sau khi bạn đóng phiên làm việc nên thông tin đó sẽ bị mất vĩnh viễn.

Bên cạnh đó, các trang web và các bên khác trên Internet vẫn có thể sử dụng nhiều kỹ thuật khác để theo dõi bạn. Vì vậy, chế độ riêng tư mang đến cho bạn một mức độ ẩn danh nhất định, nhưng nó chỉ hoàn toàn riêng tư trên máy tính bạn đang sử dụng. Nếu bạn đang truy cập tài khoản email hoặc trang web ngân hàng của mình từ máy tính công cộng, chẳng hạn như ở sân bay hoặc khách sạn, bạn nên chắc chắn rằng mình truy cập những trang đó bằng chế độ riêng tư của trình duyệt. Trong các tình huống khác thì nó có thể có lợi, nhưng bạn nên biết chính xác rủi ro nào phải tránh và rủi ro nào là vô hại. Bất cứ khi nào bạn sử dụng máy tính công cộng, hãy lưu ý rằng việc đó có thể tồn tại các mối đe dọa bảo mật khác như phần mềm độc hại hoặc trình ghi khóa. Hãy cẩn thận bất cứ khi nào bạn nhập thông tin cá nhân bao gồm tên người dùng và mật khẩu trên những máy tính đó, hoặc khi bạn tải xuống hoặc sao chép dữ liệu bảo mật.

Chọn đúng Mật khẩu

Một trong những tình huống khó khăn nhất mà bất kỳ người dùng nào cũng sẽ gặp phải là chọn mật khẩu an toàn cho các dịch vụ mà họ sử dụng. Chắc chắn bạn đã từng nghe nói rằng bạn không nên sử dụng các tổ hợp phổ biến như `qwerty`, `123456` hoặc `654321`, cũng như các số dễ đoán như ngày sinh hoặc mã bưu điện của bạn (hoặc người thân). Lý do là bởi vì chúng là những tổ hợp quá lộ liễu và cũng là những lựa chọn đầu tiên mà kẻ xâm nhập sẽ thử để dành quyền truy cập vào tài

khoản của bạn.

Đã có nhiều những kỹ thuật được phổ biến để tạo ra một mật khẩu an toàn. Một trong những loại nổi tiếng nhất là tạo ra một câu khiến bạn nhớ đến dịch vụ đó và chọn các chữ cái đầu tiên của mỗi từ. Giả sử ta muốn tạo một mật khẩu tốt cho Facebook. Trong trường hợp này, ta có thể nghĩ ra một câu như “Tôi sẽ rất vui nếu tôi có 1000 người bạn như Mike”. Chọn chữ cái đầu tiên của mỗi từ và mật khẩu cuối cùng sẽ là `tsrvntc1000nbnM`. Kết quả là bạn sẽ có một mật khẩu gồm 15 ký tự đủ dài để khó đoán và đồng thời cũng dễ nhớ (miễn là bạn có thể nhớ được câu và “quy tắc” để truy xuất mật khẩu).

Các câu thường dễ nhớ hơn mật khẩu, nhưng ngay cả phương pháp này cũng có những hạn chế. Ngày nay, chúng ta phải tạo mật khẩu cho rất nhiều dịch vụ và khi chúng ta sử dụng chúng với tần suất khác nhau, cuối cùng sẽ rất khó để nhớ tất cả các câu tại thời điểm chúng ta cần. Vậy chúng ta có thể làm gì? Có thể bạn sẽ trả lời rằng điều khôn ngoan nhất nên làm trong trường hợp này là tạo một vài mật khẩu tiêu chuẩn và sử dụng lại chúng trên các dịch vụ tương tự nhau?

Thật không may, đó cũng không phải là một ý hay. Có thể bạn cũng đã nghe tới việc không nên sử dụng lại cùng một mật khẩu giữa các dịch vụ khác nhau. Vấn đề khi làm như vậy là một dịch vụ cụ thể có thể làm rò rỉ mật khẩu của bạn (và việc này luôn xảy ra) và bất kỳ người nào có quyền truy cập vào dịch vụ đó sẽ cố gắng sử dụng cùng một tổ hợp email và mật khẩu trên các dịch vụ phổ biến khác trên Internet với hy vọng bạn đã làm chính xác điều này: tái sử dụng mật khẩu. Và bạn đoán xem? Trong trường hợp họ đúng, bạn sẽ gặp sự cố không chỉ trên một dịch vụ mà còn trên nhiều dịch vụ. Và chúng ta đều có xu hướng nghĩ rằng điều đó sẽ không xảy ra với mình cho đến khi đã quá muộn.

Nếu vậy, chúng ta có thể làm gì để bảo vệ chính mình? Một trong những cách tiếp cận an toàn nhất hiện nay là sử dụng *trình quản lý mật khẩu*. Trình quản lý mật khẩu là một phần mềm về cơ bản sẽ lưu trữ tất cả mật khẩu và tên người dùng của bạn ở định dạng được mã hóa mà mật khẩu chính có thể giải mã được. Bằng cách này, bạn chỉ cần nhớ một mật khẩu tiêu chuẩn là được, bởi trình quản lý sẽ giữ tất cả những mật khẩu khác an toàn cho bạn.

KeePass là một trong những trình quản lý mật khẩu mã nguồn mở phong phú và nổi tiếng nhất hiện có. Nó sẽ lưu trữ mật khẩu của bạn trong một tệp được mã hóa trong hệ thống tệp của bạn. Việc phần mềm này là phần mềm mã nguồn mở là một vấn đề quan trọng đối với loại phần mềm này vì nó có thể đảm bảo rằng các phần mềm sẽ không sử dụng dữ liệu của bạn vì bất kỳ nhà phát triển nào cũng có thể kiểm tra mã và biết chính xác xem nó hoạt động như thế nào. Điều này mang lại một mức độ minh bạch mà mã độc quyền không thể đạt được. KeePass có cổng cho hầu hết các hệ điều hành, bao gồm Windows, Linux và macOS, cũng như các thiết bị di động như iOS và Android. Nó cũng bao gồm một hệ thống tiện ích có thể mở rộng chức năng vượt xa các tính năng mặc định.

Bitwarden là một giải pháp mã nguồn mở khác có cách tiếp cận tương tự; nhưng thay vì lưu trữ dữ liệu của bạn trong một tệp, nó sẽ sử dụng máy chủ đám mây. Bằng cách này, việc đồng bộ hóa tất cả các thiết bị của bạn trở nên dễ dàng hơn và mật khẩu của bạn cũng có thể được truy cập dễ dàng qua web. *Bitwarden* là một trong số ít những dự án sẽ cung cấp không chỉ máy khách mà cả máy chủ đám mây dưới dạng phần mềm mã nguồn mở. Điều này có nghĩa là bạn có thể lưu trữ phiên bản *Bitwarden* của riêng mình và cung cấp phiên bản này cho bất kỳ ai, chẳng hạn như gia đình hoặc nhân viên công ty của bạn. Điều này sẽ mang lại cho bạn sự linh hoạt cùng với toàn quyền kiểm soát cách lưu trữ và sử dụng mật khẩu của họ.

Một trong những điều quan trọng nhất cần lưu ý khi sử dụng trình quản lý mật khẩu là tạo một mật khẩu ngẫu nhiên cho từng dịch vụ khác nhau vì đăng nào bạn cũng sẽ không cần phải nhớ chúng. Nếu bạn sử dụng trình quản lý mật khẩu để lưu trữ các mật khẩu tái sử dụng hoặc dễ đoán thì việc sử dụng nó sẽ trở nên vô ích. Do đó, hầu hết các trình quản lý sẽ cung cấp cho bạn một trình tạo mật khẩu ngẫu nhiên mà bạn có thể sử dụng để tạo những mật khẩu đó cho mình.

Mã hóa

Bất cứ khi nào dữ liệu được truyền hoặc lưu trữ, bạn cần thực hiện các biện pháp phòng ngừa để đảm bảo rằng các bên thứ ba không thể truy cập được dữ liệu. Dữ liệu được truyền qua internet đi qua một loạt các bộ định tuyến và mạng nơi các bên thứ ba có thể truy cập lưu lượng mạng. Tương tự như vậy, dữ liệu được lưu trữ trên phương tiện vật lý có thể được đọc bởi bất kỳ ai sở hữu phương tiện đó. Để tránh kiểu truy cập này, các thông tin bí mật phải được mã hóa trước khi rời khỏi thiết bị máy tính.

TLS

Bảo mật tầng vận tải (TLS - Transport Layer Security) là một giao thức cung cấp lớp bảo mật thông qua các kết nối mạng bằng cách sử dụng mật mã. TLS là sự kế thừa của *Lớp cổng bảo mật* (SSL - Secure Sockets Layer) đã không còn được sử dụng nữa do các sai sót nghiêm trọng của nó. TLS cũng đã được biến đổi một vài lần để có thể tự điều chỉnh và trở nên an toàn hơn, vì vậy phiên bản hiện tại của nó là 1.3. Nó có thể cung cấp cả quyền riêng tư và tính xác thực bằng cách sử dụng mật mã đối xứng và khóa công khai. Khi nói như vậy, ta có thể hiểu rằng sau khi sử dụng nó, ta có thể chắc chắn rằng không ai có thể nghe lén hoặc thay đổi giao tiếp của bạn với máy chủ trong phiên làm việc đó.

Bài học quan trọng nhất ở đây là việc biết được thế nào một trang web đáng tin cậy. Bạn nên tìm biểu tượng “khóa” trên thanh địa chỉ của trình duyệt. Nếu muốn, bạn có thể nhấp vào nó để kiểm tra chứng chỉ đóng vai trò quan trọng trong giao thức HTTPS.

TLS được sử dụng trên giao thức HTTPS (*HTTP trên TLS*) để có thể gửi dữ liệu nhạy cảm (như số thẻ tín dụng của bạn) qua web. Việc giải thích cách thức hoạt động của TLS sẽ vượt qua khuôn khổ

mục tiêu của bài học này, nhưng bạn có thể tìm thêm thông tin trên [Wikipedia](#) và [Mozilla wiki](#).

Mã hóa Tập và E-mail bằng GnuPG

Có rất nhiều các công cụ để bảo mật email, nhưng một trong những công cụ quan trọng nhất chắc chắn phải là *GnuPG*. GnuPG là viết tắt của GNU Privacy Guard (Bảo vệ quyền riêng tư GNU) và nó là một phần triển khai của *OpenPGP* - một tiêu chuẩn quốc tế được hệ thống hóa trong RFC 4880.

GnuPG có thể được sử dụng để ký tên, mã hóa và giải mã văn bản, thư điện tử, tệp, thư mục và thậm chí là toàn bộ phân vùng đĩa. Nó hoạt động với mật mã khóa công khai và được phổ biến rộng rãi. Tóm lại, GnuPG sẽ tạo một cặp tệp chứa khóa công khai và khóa riêng tư của bạn. Đúng như tên gọi, khóa công khai có thể được cung cấp cho bất kỳ ai và khóa riêng tư thì cần phải được giữ bí mật. Mọi người sẽ sử dụng khóa công khai của bạn để mã hóa dữ liệu mà chỉ khóa riêng tư của bạn mới có thể giải mã được.

Bạn cũng có thể sử dụng khóa riêng tư của mình để ký bất kỳ tệp hoặc thư điện tử nào có thể được xác thực đối với khóa công khai tương ứng. Chữ ký kỹ thuật số này hoạt động tương tự như chữ ký trong thế giới thực. Miễn là bạn là người duy nhất sở hữu khóa riêng tư của mình, người nhận có thể chắc chắn rằng chính bạn là người đã tạo ra nó. Bằng cách sử dụng chức năng băm mật mã, GnuPG cũng sẽ đảm bảo không có thay đổi nào được thực hiện sau chữ ký vì bất kỳ thay đổi nào đối với nội dung sẽ làm mất đi hiệu lực của chữ ký.

GnuPG là một công cụ rất mạnh mẽ và ở một mức độ nào đó cũng là một công cụ phức tạp. Bạn có thể tìm thêm thông tin trên [trang web](#) và [Archlinux wiki](#) (Archlinux wiki là một nguồn thông tin rất tốt ngay cả khi bạn không sử dụng Archlinux).

Mã hóa Đĩa

Một cách tốt để bảo mật dữ liệu của bạn là mã hóa toàn bộ đĩa hoặc phân vùng đĩa của bạn. Có nhiều phần mềm mã nguồn mở mà bạn có thể sử dụng để đạt được mục đích này. Cách chúng hoạt động và mức độ mã hóa mà chúng cung cấp cũng khác nhau đáng kể. Có hai phương pháp cơ bản: mã hóa thiết bị *xếp chồng* và mã hoá thiết bị *khối*.

Các giải pháp hệ thống tệp xếp chồng sẽ được triển khai trên hệ thống tệp hiện có. Khi sử dụng phương pháp này, các tệp và thư mục sẽ được mã hóa trước khi được lưu trữ trên hệ thống tệp và được giải mã sau khi đọc chúng. Điều này có nghĩa là các tệp sẽ được lưu trữ trên hệ thống tệp máy chủ ở dạng được mã hóa (có nghĩa là nội dung của chúng và thường là cả tên tệp/thư mục cũng sẽ được thay thế bằng dữ liệu tìm kiếm ngẫu nhiên); ngoài ra, chúng vẫn tồn tại trong hệ thống tệp đó như khi không được mã hóa giống như các tệp, liên kết tượng trưng, liên kết cứng, v.v. thông thường.

Mặt khác, mã hóa thiết bị khối xảy ra ở bên dưới lớp hệ thống tệp, đảm bảo mọi thứ được ghi vào

thiết bị khối đều sẽ được mã hóa. Nếu bạn nhìn vào khối khi nó ngoại tuyến (offline), nó sẽ trông giống như một phần lớn dữ liệu ngẫu nhiên và bạn thậm chí sẽ không thể biết được loại hệ thống tệp nào có ở đó nếu không giải mã nó trước. Điều này có nghĩa là bạn sẽ không thể biết nó là tệp hay thư mục là gì, kích cỡ ra sao và đó là loại dữ liệu gì vì siêu dữ liệu, cấu trúc thư mục và quyền truy cập đều được mã hóa.

Cả hai phương pháp đều có ưu và nhược điểm riêng. Trong số tất cả các tùy chọn có sẵn, bạn nên thử *dm-crypt*. Đây là tiêu chuẩn thực tế để mã hóa khối cho các hệ thống Linux vì nó có sẵn trong nhân. Nó có thể được sử dụng với phần mở rộng *LUKS* (*Linux Unified Key Setup* - Thiết lập khóa hợp nhất Linux) vốn là một đặc điểm kỹ thuật thực thi một tiêu chuẩn độc lập với nền tảng để sử dụng với các công cụ khác nhau.

Nếu bạn muốn thử phương pháp xếp chồng, bạn nên thử *EncFS*; đây có lẽ là cách dễ nhất để bảo mật dữ liệu trên Linux vì nó không yêu cầu quyền gốc để triển khai và có thể hoạt động trên một hệ thống tệp có sẵn mà không cần sửa đổi.

Cuối cùng, nếu bạn cần truy cập dữ liệu trên nhiều nền tảng khác nhau, hãy thử Veracrypt. Nó là sự kế thừa của Truecrypt và cho phép tạo các phương tiện và tệp được mã hóa; chúng có thể được sử dụng trên Linux cũng như trên macOS và Windows.

Bài tập Hướng dẫn

1. Bạn nên sử dụng “cửa sổ riêng tư” trong trình duyệt của mình nếu bạn muốn:

Truy cập ẩn danh hoàn toàn trên Internet	
Không để lại dấu vết trên máy tính bạn đang sử dụng	
Kích hoạt TLS để tránh cookie theo dõi	
Sử dụng DNT	
Sử dụng mật mã trong quá trình truyền dữ liệu	

2. OpenStack là gì?

Một dự án cho phép tạo IaaS riêng tư	
Một dự án cho phép tạo PaaS riêng tư	
Một dự án cho phép tạo SaaS riêng tư	
Một trình giám sát máy ảo	
Trình quản lý mật khẩu mã nguồn mở	

3. Tùy chọn nào dưới đây là phần mềm mã hóa đĩa hợp lệ?

RevealJS, EncFS và dm-crypt	
dm-crypt và KeePass	
EncFS và Bitwarden	
EncFS và dm-crypt	
TLS và dm-crypt	

4. Chọn Đúng hoặc Sai đối với mã hóa thiết bị dm-crypt:

Các tệp được mã hóa trước khi ghi vào đĩa	
Toàn bộ hệ thống tệp sẽ là một đối tượng nhị phân lớn được mã hóa	
Chỉ các tệp và thư mục mới được mã hóa chứ không phải liên kết tượng trưng	

Không yêu cầu quyền gốc	
Là một mã hóa thiết bị khối	

5. Beamer là:

Một cơ chế mã hóa	
Một trình giám sát máy ảo	
Một phần mềm ảo hoá	
Một thành phần của OpenStack	
Một công cụ trình chiếu LaTeX	

Bài tập Mở rộng

1. Hầu hết các bản phân phối đều được cài đặt Firefox theo mặc định (nếu bản của bạn không có, bạn sẽ phải cài đặt nó trước). Chúng ta đang chuẩn bị cài đặt một tiện ích mở rộng của Firefox có tên *Lightbeam*. Bạn có thể làm điều này bằng cách nhấn tổ hợp `Ctrl` + `Shift` + `A` và tìm kiếm “Lightbeam” trên trường tìm kiếm được hiển thị trên tab đã mở hoặc bằng cách truy cập trang tiện ích mở rộng bằng Firefox và nhấp vào " Nút `Cài đặt`": <https://addons.mozilla.org/en-GB/firefox/addon/lightbeam-3-0/>. Sau khi thực hiện việc này, hãy khởi động tiện ích mở rộng bằng cách nhấp vào biểu tượng của nó và bắt đầu truy cập một số trang web trên các tab khác để xem điều gì sẽ xảy ra.

2. Điều quan trọng nhất khi sử dụng trình quản lý mật khẩu là gì?

3. Sử dụng trình duyệt web của bạn để điều hướng đến <https://haveibeenpwned.com/>. Tìm hiểu mục đích của trang web và kiểm tra xem địa chỉ email của bạn có bị rò rỉ dữ liệu hay không.

Tóm tắt

Cửa sổ dòng lệnh là một cách hữu hiệu để tương tác với hệ thống với rất nhiều công cụ hữu ích và hoàn thiện để sử dụng trong loại môi trường này. Bạn có thể truy cập cửa sổ dòng lệnh bằng cách tìm kiếm biểu tượng đồ họa của nó trong menu môi trường máy tính để bàn hoặc nhấn tổ hợp **Ctrl + Alt + F#**.

Linux chủ yếu được sử dụng trong ngành công nghệ để cung cấp các dịch vụ IaaS, PaaS và SaaS. Có ba trình giám sát máy ảo chính đóng vai trò quan trọng trong việc hỗ trợ chúng: Xen, KVM và Virtualbox.

Trình duyệt là một phần mềm thiết yếu trong máy tính ngày nay, nhưng có một số điều người dùng cần hiểu rõ để có thể sử dụng nó một cách an toàn. DNT chỉ là một cách để nói với trang web rằng bạn không muốn bị theo dõi, nhưng không có gì đảm bảo cho điều đó. Các cửa sổ riêng tư tuy chỉ dành riêng cho máy tính bạn đang sử dụng nhưng chúng có thể cho phép bạn thoát khỏi cookie theo dõi chính vì lẽ này.

TLS có thể mã hóa thông tin liên lạc của bạn trên Internet, nhưng bạn phải có khả năng nhận ra khi nó được sử dụng. Sử dụng mật khẩu mạnh cũng rất quan trọng để giữ an toàn cho bạn, vì vậy tốt nhất là giao trách nhiệm đó cho trình quản lý mật khẩu và cho phép phần mềm đó tạo ra mật khẩu ngẫu nhiên cho mọi trang web bạn đăng nhập.

Một cách khác để bảo mật thông tin liên lạc của bạn là ký và mã hóa các thư mục tệp và email của bạn bằng GnuPG. dm-crypt và EncFS là hai lựa chọn thay thế tương ứng để mã hóa toàn bộ đĩa hoặc các phân vùng sử dụng các phương thức mã hóa khối và xếp chồng.

Cuối cùng, LibreOffice Impress là một giải pháp mã nguồn mở thay thế rất hoàn chỉnh cho Microsoft Powerpoint; nhưng ngoài ra cũng có Beamer và RevealJS nếu bạn muốn tạo bản trình chiếu bằng mã thay vì GUI. ProjectLibre và GanttProject cũng có thể là những lựa chọn phù hợp nếu bạn cần thay thế Microsoft Project.

Đáp án Bài tập Hướng dẫn

1. Bạn nên sử dụng “cửa sổ riêng tư” trong trình duyệt của mình nếu bạn muốn:

Truy cập ẩn danh hoàn toàn trên Internet	
Không để lại dấu vết trên máy tính bạn đang sử dụng	X
Kích hoạt TLS để tránh cookie theo dõi	
Sử dụng DNT	
Sử dụng mật mã trong quá trình truyền dữ liệu	

2. OpenStack là gì?

Một dự án cho phép tạo IaaS riêng tư	X
Một dự án cho phép tạo PaaS riêng tư	
Một dự án cho phép tạo SaaS riêng tư	
Một trình giám sát máy ảo	
Trình quản lý mật khẩu mã nguồn mở	

3. Tùy chọn nào dưới đây là phần mềm mã hóa đĩa hợp lệ?

RevealJS, EncFS và dm-crypt	
dm-crypt và KeePass	
EncFS và Bitwarden	
EncFS và dm-crypt	X
TLS và dm-crypt	

4. Chọn Đúng hoặc Sai đối với mã hóa thiết bị dm-crypt:

Các tệp được mã hóa trước khi ghi vào đĩa	Đ
Toàn bộ hệ thống tệp là một đối tượng nhị phân lớn được mã hóa	Đ
Chỉ các tệp và thư mục được mã hóa, không phải liên kết tượng trưng	S

Không yêu cầu quyền gốc	S
Là một mã hóa thiết bị khối	Đ

5. Beamer là:

Một cơ chế mã hóa	
Một trình giám sát máy ảo	
Một phần mềm ảo hoá	
Một thành phần của OpenStack	
Một công cụ trình chiếu LaTeX	X

Đáp án Bài tập Mở rộng

1. Hầu hết các bản phân phối đều được cài đặt Firefox theo mặc định (nếu bản của bạn không có, bạn sẽ phải cài đặt nó trước). Chúng ta đang chuẩn bị cài đặt một tiện ích mở rộng của Firefox có tên *Lightbeam*. Bạn có thể làm điều này bằng cách nhấn tổ hợp `Ctrl` + `Shift` + `A` và tìm kiếm “Lightbeam” trên trường tìm kiếm được hiển thị trên tab đã mở hoặc bằng cách truy cập trang tiện ích mở rộng bằng Firefox và nhấp vào " Nút `Cài đặt`": <https://addons.mozilla.org/en-GB/firefox/addon/lightbeam-3-0/>. Sau khi thực hiện việc này, hãy khởi động tiện ích mở rộng bằng cách nhấp vào biểu tượng của nó và bắt đầu truy cập một số trang web trên các tab khác để xem điều gì sẽ xảy ra.

Bạn có nhớ những cookie đã được đề cập tới ở trên có thể chia sẻ dữ liệu của bạn với nhiều dịch vụ khác nhau khi bạn truy cập một trang web không? Đó chính là những gì tiện ích mở rộng này sẽ cho bạn thấy. Lightbeam là một thử nghiệm của Mozilla nhằm cho bạn biết các trang web của bên thứ nhất và bên thứ ba mà bạn tương tác khi truy cập một URL. Nội dung này thường không hiển thị đối với người dùng bình thường và nó có thể cho thấy rằng đôi khi một trang web có thể tương tác với hàng chục dịch vụ khác nhau trở lên.

2. Điều quan trọng nhất khi sử dụng trình quản lý mật khẩu là gì?

Khi sử dụng trình quản lý mật khẩu, điều quan trọng nhất cần lưu ý là ghi nhớ mật khẩu chính của bạn và sử dụng các mật khẩu ngẫu nhiên độc nhất cho từng dịch vụ khác nhau.

3. Sử dụng trình duyệt web của bạn để điều hướng đến <https://haveibeenpwned.com/>. Tìm hiểu mục đích của trang web và kiểm tra xem địa chỉ email của bạn có bị rò rỉ dữ liệu hay không.

Trang web này duy trì một cơ sở dữ liệu các thông tin đăng nhập có mật khẩu bị ảnh hưởng do rò rỉ mật khẩu. Nó cho phép ta tìm kiếm địa chỉ email và cho biết nếu địa chỉ email đó đã được đưa vào cơ sở dữ liệu công khai về thông tin đăng nhập bị đánh cắp. Rất có thể địa chỉ email của bạn cũng bị ảnh hưởng bởi một hoặc nhiều vụ rò rỉ khác. Nếu đúng như vậy, hãy đảm bảo rằng bạn đã cập nhật mật khẩu của mình trong thời gian gần đây. Nếu bạn chưa sử dụng trình quản lý mật khẩu, hãy xem những ví dụ đã được đề xuất trong bài học này.



**Linux
Professional
Institute**

Chủ đề 2: Tìm một lối đi trong Hệ thống Linux



2.1 Khái niệm cơ bản về dòng lệnh

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 2.1](#)

Khối lượng

3

Các lĩnh vực kiến thức chính

- Vô cơ bản
- Cú pháp dòng lệnh
- Biến
- Trích dẫn

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

- Bash
- echo
- history
- PATH environment variable
- export
- type



2.1 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	2 Tìm đường trong Hệ thống Linux
Mục tiêu:	2.1 Những Khái niệm cơ bản về Dòng Lệnh
Bài:	1 trên 2

Giới thiệu

Các bản phân phối Linux hiện đại có rất nhiều loại giao diện đồ họa người dùng, nhưng quản trị viên luôn phải biết cách làm việc với dòng lệnh hay còn gọi là vỏ (shell). Vỏ là một chương trình cho phép giao tiếp dựa trên văn bản giữa hệ điều hành và người dùng. Nó thường là một chương trình ở chế độ văn bản và có thể đọc đầu vào của người dùng rồi diễn giải nó dưới dạng các lệnh cho hệ thống.

Có một số trình vỏ khác nhau trên Linux, dưới đây là một số ví dụ:

- Vỏ Bourne-again (Bash)
- Vỏ C (csh hoặc tcsh, csh nâng cao)
- Vỏ Korn (ksh)
- Vỏ Z (zsh)

Trên Linux, vỏ phổ biến nhất là Vỏ Bash. Đây cũng là vỏ sẽ được sử dụng trong các ví dụ hoặc bài tập trong bài học này.

Khi sử dụng một vỏ tương tác, người dùng sẽ nhập lệnh tại dấu nhắc lệnh. Dấu nhắc lệnh mặc

định có thể sẽ hơi khác nhau một chút tùy theo từng bản phân phối của Linux, nhưng chúng thường đều tuân theo cấu trúc sau:

```
username@hostname current_directory shell_type
```

Trên Ubuntu hoặc Debian GNU/Linux, dấu nhắc lệnh dành cho người dùng thông thường sẽ có dạng như sau:

```
carol@mycomputer:~$
```

Dấu nhắc lệnh của siêu người dùng sẽ giống như sau:

```
root@mycomputer:~#
```

Trên CentOS hoặc Red Hat Linux, dấu nhắc lệnh cho người dùng thông thường sẽ giống như sau:

```
[dave@mycomputer ~]$
```

Dấu nhắc lệnh của siêu người dùng sẽ giống như sau:

```
[root@mycomputer ~]#
```

Hãy cùng giải thích từng thành phần của cấu trúc này:

username

Tên của người dùng chạy vỏ

hostname

Tên của máy chủ mà vỏ chạy trên đó. Ngoài ra còn có một lệnh `hostname` mà bạn có thể dùng để hiển thị hoặc đặt tên máy chủ của hệ thống.

current_directory

Thư mục chứa vỏ hiện tại. `~` có nghĩa là vỏ đang nằm trong thư mục chính của người dùng hiện tại.

shell_type

`$` cho biết vỏ được điều hành bởi người dùng thông thường.

cho biết vỏ được điều hành bởi siêu người dùng `root`.

Vì không cần bất kỳ đặc quyền đặc biệt nào nên chúng ta sẽ sử dụng một dấu nhắc lệnh không có đặc quyền trong các ví dụ tiếp theo. Để ngắn gọn hơn, ta sẽ chỉ sử dụng `$` làm dấu nhắc lệnh.

Cấu trúc của Dòng lệnh

Hầu hết các lệnh tại dòng lệnh đều sẽ tuân theo một cấu trúc cơ bản giống nhau:

```
command [option(s)/parameter(s)...] [argument(s)...]
```

Lấy lệnh sau làm ví dụ:

```
$ ls -l /home
```

Hãy cùng giải thích mục đích của từng thành phần:

Command (Lệnh)

Chương trình mà người dùng sẽ chạy – trong ví dụ trên là `ls`.

Option(s)/Parameter(s) (Tùy chọn/ Tham số)

Một “khóa chuyển” dùng để sửa đổi hành vi của lệnh theo một cách nào đó, chẳng hạn như `-l` trong ví dụ trên. Các tùy chọn có thể được truy cập ở cả dạng ngắn và dài. Ví dụ: `-l` sẽ giống với `--format=long`.

Có thể kết hợp nhiều tùy chọn với nhau; đối với dạng viết tắt, các chữ cái thường có thể được nhập cùng nhau. Ví dụ: tất cả các lệnh sau đều thực hiện cùng một việc:

```
$ ls -al
$ ls -a -l
$ ls --all --format=long
```

Argument(s) (Đối số)

Dữ liệu bổ sung mà chương trình yêu cầu, ví dụ như tên tệp hoặc đường dẫn (như `/home` trong ví dụ trên).

Phần bắt buộc duy nhất của cấu trúc này là lệnh. Thông thường, tất cả các phần khác đều là tùy chọn, nhưng một chương trình có thể yêu cầu chỉ định một số tùy chọn, tham số hoặc đối số cụ thể.

NOTE

Hầu hết các lệnh đều sẽ hiển thị một tổng quan ngắn về các tùy chọn khả dụng khi chúng được thực thi với tham số `--help`. Chúng ta sẽ sớm được học thêm về các cách khác để tìm hiểu sâu hơn về các lệnh trong Linux.

Các loại Hành vi của Lệnh

Vỏ hỗ trợ hai loại lệnh:

Internal (Lệnh nội bộ)

Những lệnh này là một phần của chính vỏ và không phải là các chương trình riêng biệt. Có khoảng 30 lệnh như vậy. Mục đích chính của chúng là thực thi các tác vụ bên trong vỏ (ví dụ: `cd`, `set`, `export`).

External (Lệnh bên ngoài)

Các lệnh này nằm trong các tệp riêng lẻ. Các tệp này thường là các chương trình nhị phân hoặc tệp lệnh. Khi một lệnh không phải là lệnh tích hợp sẵn của vỏ được chạy, vỏ sẽ sử dụng biến `PATH` để tìm kiếm một tệp có thể thực thi có cùng tên với lệnh. Ngoài các chương trình được cài đặt cùng với trình quản lý gói của bản phân phối, người dùng cũng có thể tạo các lệnh bên ngoài của riêng họ.

Lệnh `type` cho biết cụ thể đó là loại lệnh nào:

```
$ type echo
echo is a shell builtin
$ type man
man is /usr/bin/man
```

Trích dẫn

Là người dùng Linux, bạn sẽ phải tạo hoặc thao tác với các tệp hoặc các biến theo nhiều cách khác nhau. Điều này khá là dễ dàng khi làm việc với các tên tệp ngắn hay các giá trị đơn lẻ, nhưng sẽ trở nên phức tạp hơn khi có liên quan đến chẳng hạn như khoảng trắng, ký tự đặc biệt và biến. Vỏ cung cấp một tính năng được gọi là trích dẫn để gói gọn các dữ liệu đó bằng nhiều loại trích dẫn khác nhau (" ", ' '). Có ba loại trích dẫn trong Bash:

- Trích dẫn kép
- Trích dẫn đơn
- Ký tự Thoát

Ví dụ: các lệnh sau không hoạt động theo cùng một kiểu do các cách trích dẫn của chúng khác nhau:

```
$ TWOWORDS="two words"
$ touch $TWOWORDS
$ ls -l
-rw-r--r-- 1 carol carol    0 Mar 10 14:56 two
-rw-r--r-- 1 carol carol    0 Mar 10 14:56 words
$ touch "$TWOWORDS"
$ ls -l
-rw-r--r-- 1 carol carol    0 Mar 10 14:56 two
-rw-r--r-- 1 carol carol    0 Mar 10 14:58 'two words'
-rw-r--r-- 1 carol carol    0 Mar 10 14:56 words
$ touch '$TWOWORDS'
$ ls -l
-rw-r--r-- 1 carol carol    0 Mar 10 15:00 '$TWOWORDS'
-rw-r--r-- 1 carol carol    0 Mar 10 14:56 two
-rw-r--r-- 1 carol carol    0 Mar 10 14:58 'two words'
-rw-r--r-- 1 carol carol    0 Mar 10 14:56 words
```

NOTE

Dòng có `TWOWORDS=` là một biến Bash đã được tạo. Định nghĩa về biến sẽ được đề cập đến sau. Ví dụ này chỉ nhằm cho bạn thấy việc trích dẫn ảnh hưởng đến đầu ra của các biến như thế nào.

Trích dẫn Kép

Dấu trích dẫn kép ("...") cho phép vỏ xử lý văn bản ở giữa nó như những ký tự thông thường. Tất cả các ký tự đặc biệt đều sẽ mất ý nghĩa, ngoại trừ `$` (ký hiệu đô la), `\` (dấu gạch chéo ngược) và ``` (trích dẫn ngược). Điều này có nghĩa là các biến, các trình thay thế lệnh và các hàm số học vẫn có thể được sử dụng.

Ví dụ: việc thay thế biến `$USER` không bị ảnh hưởng bởi dấu trích dẫn kép:

```
$ echo I am $USER
I am tom
$ echo "I am $USER"
I am tom
```

Mặt khác, một ký tự khoảng trắng sẽ không còn là một dấu phân tách đối số nữa:

```
$ touch new file
```

```
$ ls -l
-rw-rw-r-- 1 tom students 0 Oct 8 15:18 file
-rw-rw-r-- 1 tom students 0 Oct 8 15:18 new
$ touch "new file"
$ ls -l
-rw-rw-r-- 1 tom students 0 Oct 8 15:19 new file
```

Như bạn thấy, trong ví dụ đầu tiên, lệnh `touch` đã tạo ra hai tệp riêng lẻ; lệnh này diễn giải hai chuỗi dưới dạng các đối số riêng lẻ. Trong ví dụ thứ hai, lệnh diễn giải cả hai chuỗi dưới dạng một đối số; do đó, nó chỉ tạo ra một tệp. Tuy nhiên, tốt nhất là nên tránh ký tự khoảng trắng trong tên tệp. Thay vào đó, ta có thể sử dụng dấu gạch dưới (`_`) hoặc dấu chấm (`.`).

Trích dẫn Đơn

Dấu trích dẫn đơn không có ngoại lệ của dấu trích dẫn kép. Chúng sẽ loại bỏ bất kỳ ý nghĩa đặc biệt nào của mỗi một ký tự. Hãy lấy một trong những ví dụ đầu tiên đã nêu ở trên:

```
$ echo I am $USER
I am tom
```

Khi áp dụng các trích dẫn đơn, bạn sẽ thấy một kết quả khác:

```
$ echo 'I am $USER'
I am $USER
```

Bây giờ lệnh lại hiện hiển thị chuỗi chính xác mà không cần thay thế biến.

Ký tự Thoát

Chúng ta có thể sử dụng *ký tự thoát* để xóa đi ý nghĩa đặc biệt của các ký tự khỏi Bash. Quay lại biến môi trường `$USER`:

```
$ echo $USER
carol
```

Ta có thể thấy rằng, theo mặc định, nội dung của biến sẽ được hiển thị trong cửa sổ dòng lệnh. Tuy nhiên, nếu chúng ta sử dụng ký tự gạch chéo (`\`) trước ký hiệu đô la (`$`) thì ý nghĩa đặc biệt của ký hiệu đô la sẽ bị bỏ qua. Điều này sẽ không cho phép Bash mở rộng giá trị của biến thành tên người dùng đang chạy lệnh, mà thay vào đó sẽ giải thích tên biến theo nghĩa đen:

```
$ echo \USER
USER
```

Nếu bạn còn nhớ, ta có thể nhận được các kết quả tương tự như kết quả này bằng cách sử dụng trích dẫn đơn để cho ra nội dung nguyên bản của bất kỳ thứ gì nằm giữa các dấu nháy đơn. Tuy nhiên, ký tự thoát hoạt động bằng cách chỉ dẫn Bash bỏ qua bất kỳ ý nghĩa đặc biệt nào mà ký tự đứng trước nó có thể có.

Bài tập Hướng dẫn

1. Hãy chia các dòng bên dưới thành các thành phần của lệnh, tùy chọn/ tham số và đối số:

- Ví dụ: `cat -n /etc/passwd`

Lệnh:	<code>cat</code>
Tùy chọn:	<code>-n</code>
Đối số:	<code>/etc/passwd</code>

- `ls -l /etc`

Lệnh:	
Tùy chọn:	
Đối số:	

- `ls -l -a`

Lệnh:	
Tùy chọn:	
Đối số:	

- `cd /home/user`

Lệnh:	
Tùy chọn:	
Đối số:	

2. Hãy cho biết các lệnh sau thuộc loại lệnh nào:

Ví dụ:

<code>pwd</code>	Lệnh tích hợp sẵn của vỏ
<code>mv</code>	Lệnh bên ngoài

<code>cd</code>	
<code>cat</code>	

<code>exit</code>	
-------------------	--

3. Hãy giải quyết các lệnh sau sử dụng trích dẫn kép:

Ví dụ:

<code>echo "\$HOME is my home directory"</code>	
---	--

<code>echo /home/user is my home directory</code>

<code>touch "\$USER"</code>	
-----------------------------	--

<code>touch 'touch'</code>	
----------------------------	--

Bài tập Mở rộng

1. Hãy sử dụng một lệnh và phần mở rộng dấu ngoặc trong Bash (xem lại trang hướng dẫn về Bash), hãy tạo 5 tệp được đánh số từ 1 đến 5 với tiền tố game (game1, game2, ...).

2. Hãy xóa tất cả 5 tệp mà bạn vừa tạo chỉ với một lệnh và một ký tự đặc biệt khác (xem lại phần *Mở rộng Tên đường dẫn* trong trang hướng dẫn của Bash).

3. Có cách nào khác để làm cho hai lệnh tương tác với nhau không? Đó là những cách gì?

Tóm tắt

Trong bài học này, bạn đã học về:

- Các khái niệm về vỏ Linux
- Vỏ Bash là gì
- Cấu trúc của Dòng lệnh
- Khái niệm cơ bản về Trích dẫn

Các lệnh được dùng trong bài tập:

bash

Vỏ phổ biến nhất trên máy tính Linux.

echo

Xuất văn bản trên cửa sổ dòng lệnh.

ls

Liệt kê nội dung của một thư mục.

type

Chỉ ra cách một lệnh cụ thể được thực thi.

touch

Tạo một tệp trống hoặc cập nhật ngày sửa đổi của tệp hiện có.

hostname

Hiển thị hoặc thay đổi tên máy chủ của hệ thống.

Đáp án Bài tập Hướng dẫn

1. Hãy chia các dòng bên dưới thành các thành phần của lệnh, tùy chọn/ tham số và đối số:

- `ls -l /etc`

Lệnh:	<code>ls</code>
Tùy chọn:	<code>-l</code>
Đối số:	<code>/etc</code>

- `ls -l -a`

Lệnh:	<code>ls</code>
Tùy chọn:	<code>-l -a</code>
Đối số:	

- `cd /home/user`

Lệnh:	<code>cd</code>
Tùy chọn:	
Đối số:	<code>/home/user</code>

2. Hãy cho biết các lệnh sau thuộc loại lệnh nào:

<code>cd</code>	Lệnh tích hợp sẵn của vỏ
<code>cat</code>	Lệnh bên ngoài
<code>exit</code>	Lệnh tích hợp sẵn của vỏ

3. Hãy giải quyết các lệnh sau sử dụng trích dẫn kép:

<code>touch "\$USER"</code>	<code>tom</code>
<code>touch 'touch'</code>	Creates a file named <code>touch</code>

Đáp án Bài tập Mở rộng

1. Hãy sử dụng một lệnh và phần mở rộng dấu ngoặc trong Bash (xem lại trang hướng dẫn về Bash), hãy tạo 5 tệp được đánh số từ 1 đến 5 với tiền tố `game` (`game1`, `game2`, ...).

Phạm vi có thể được sử dụng để thể hiện các số từ 1 đến 5 trong một lệnh:

```
$ touch game{1..5}
$ ls
game1 game2 game3 game4 game5
```

2. Hãy xóa tất cả 5 tệp mà bạn vừa tạo chỉ với một lệnh và một ký tự đặc biệt khác (xem lại phần *Mở rộng Tên đường dẫn* trong trang hướng dẫn của Bash).

Vì tất cả các tệp đều bắt đầu bằng `game` và kết thúc bằng một ký tự đơn (số từ 1 đến 5 trong trường hợp này) nên ? có thể được sử dụng làm ký tự đặc biệt cho ký tự cuối cùng trong tên tệp:

```
$ rm game?
```

3. Có cách nào khác để làm cho hai lệnh tương tác với nhau không? Đó là những cách gì?

Có. Ví dụ, một lệnh có thể ghi dữ liệu vào một tệp, tệp này sau đó sẽ được xử lý bởi một lệnh khác. Linux cũng có thể thu thập đầu ra của một lệnh và sử dụng nó làm đầu vào cho một lệnh khác. Điều này được gọi là *dẫn ống* (piping) và chúng ta sẽ tìm hiểu thêm về nó trong bài học sau.



2.1 Bài 2

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	2 Tìm đường trong Hệ thống Linux
Mục tiêu:	2.1 Những khái niệm cơ bản về Dòng lệnh
Bài:	2 trên 2

Giới thiệu

Tất cả các vỏ đều quản lý một tập hợp các thông tin trạng thái trong suốt các phiên làm việc của nó. Những thông tin thời gian chạy này có thể thay đổi trong phiên làm việc và ảnh hưởng đến cách hoạt động của vỏ. Dữ liệu này cũng được các chương trình sử dụng để xác định các khía cạnh của cấu hình hệ thống. Hầu hết những dữ liệu này sẽ được lưu trữ trong cái được gọi là *biến* mà chúng ta sẽ đề cập tới trong bài học này.

Biến

Biến là các phần lưu trữ dữ liệu, chẳng hạn như văn bản hoặc số liệu. Một khi được thiết lập, giá trị của một biến có thể được truy cập sau đó. Mỗi biến đều có một cái tên để truy cập cụ thể, ngay cả khi nội dung của biến đó thay đổi. Đây là một công cụ rất phổ biến trong hầu hết các ngôn ngữ lập trình.

Có hai loại biến trong hầu hết các vỏ Linux:

Biến cục bộ

Các biến này chỉ khả dụng cho quá trình vỏ hiện tại. Nếu bạn tạo một biến cục bộ và sau đó bắt

đầu một chương trình khác từ vỏ này, biến đó sẽ không thể truy cập được bởi chương trình đó nữa. Bởi các quy trình con không kế thừa chúng nên các biến này được gọi là *biến cục bộ*.

Biến môi trường

Các biến này có sẵn trong cả một phiên vỏ cụ thể lẫn trong các quy trình con được sinh ra từ phiên vỏ đó. Chúng có thể được sử dụng để truyền dữ liệu cấu hình cho các lệnh đang chạy. Vì các chương trình này có thể truy cập các biến này nên chúng được gọi là *biến môi trường*. Phần lớn các biến môi trường sẽ được viết bằng chữ in hoa (ví dụ như PATH, DATE, USER). Ví dụ, một tập hợp các biến môi trường mặc định sẽ cung cấp thông tin về thư mục chính của người dùng hoặc loại cửa sổ dòng lệnh. Đôi khi một tập hợp đầy đủ tất cả các biến môi trường còn được gọi là *môi trường*.

NOTE

Các biến sẽ không có mãi. Khi vỏ chứa chúng được đóng lại, tất cả các biến và nội dung của chúng sẽ bị mất. Hầu hết các vỏ đều sẽ cung cấp các tệp cấu hình có chứa các biến được thiết lập bất cứ khi nào một vỏ mới được khởi động. Những biến cần phải được thiết lập vĩnh viễn phải được thêm vào một trong những tệp cấu hình này.

Thao tác với Biến

Là quản trị viên hệ thống, bạn sẽ cần thao tác tạo, sửa đổi hoặc xóa cả biến cục bộ lẫn biến môi trường.

Làm việc với Biến Cục bộ

Bạn có thể thiết lập một biến cục bộ bằng cách sử dụng toán tử = (dấu bằng). Một phép gán đơn giản sẽ tạo ra được một biến cục bộ:

```
$ greeting=hello
```

NOTE

Không đặt bất kỳ khoảng trắng nào trước hoặc sau toán tử =.

Bạn có thể hiển thị bất kỳ biến nào bằng lệnh echo. Lệnh này thường sẽ hiển thị văn bản trong phần đối số:

```
$ echo greeting
greeting
```

Để truy cập giá trị của biến, bạn sẽ cần sử dụng \$ (ký hiệu đô la) trước tên của biến.

```
$ echo $greeting
hello
```

Như có thể thấy, biến đã được tạo. Bây giờ, hãy mở một vỏ khác và thử hiển thị nội dung của biến đã tạo.

```
$ echo $greeting
```

Không có gì được hiển thị. Từ đây ta có thể hiểu là các biến luôn chỉ tồn tại trong một vỏ cụ thể.

Để xác minh rằng biến đó thực sự là một biến cục bộ, hãy thử tạo một quy trình mới và kiểm tra xem quy trình này có thể truy cập biến đó hay không. Chúng ta có thể làm việc này bằng cách khởi động một vỏ khác và để vỏ này chạy lệnh `echo`. Vì vỏ mới được chạy trong một quy trình mới nên nó sẽ không kế thừa các biến cục bộ từ quy trình mẹ của nó:

```
$ echo $greeting world
hello world
$ bash -c 'echo $greeting world'
world
```

NOTE | Hãy nhớ sử dụng trích dẫn đơn trong ví dụ trên.

Để xóa một biến, bạn sẽ cần sử dụng lệnh `unset`:

```
$ echo $greeting
hey
$ unset greeting
$ echo $greeting
```

NOTE | `unset` sẽ yêu cầu tên của biến làm đối số. Do đó, bạn không thể thêm `$` vào tên vì điều này sẽ giải quyết biến và sẽ chuyển giá trị của biến thành `unset` thay vì tên của biến.

Làm việc với Biến Toàn cục

Để cung cấp một biến cho các quy trình con, hãy biến biến đó từ biến cục bộ thành biến môi trường. Điều này có thể được thực hiện bằng lệnh `export`. Khi nó được gọi với tên biến, biến này sẽ được thêm vào môi trường của vỏ:

```
$ greeting=hello
$ export greeting
```

NOTE

Một lần nữa, hãy nhớ không sử dụng `$` khi chạy `export` vì cái bạn muốn chuyển là tên của biến chứ không phải nội dung của nó.

Một cách dễ hơn để tạo biến môi trường là kết hợp cả hai phương pháp trên bằng cách gán giá trị của biến vào phần đối số của lệnh.

```
$ export greeting=hey
```

Hãy kiểm tra lại xem biến có thể truy cập được trong các quy trình con hay không:

```
$ export greeting=hey
$ echo $greeting world
hey world
$ bash -c 'echo $greeting world'
hey world
```

Một cách khác để sử dụng các biến môi trường là sử dụng chúng trước các lệnh. Chúng ta có thể kiểm tra điều này với biến môi trường `TZ` chứa múi giờ. Biến này có thể được sử dụng bởi lệnh `date` để xác định thời gian của múi giờ sẽ hiển thị:

```
$ TZ=EST date
Thu 31 Jan 10:07:35 EST 2019
$ TZ=GMT date
Thu 31 Jan 15:07:35 GMT 2019
```

Bạn có thể hiển thị tất cả các biến môi trường bằng lệnh `env`.

Biến PATH

Biến `PATH` là một trong những biến môi trường quan trọng nhất trong hệ thống Linux. Nó lưu trữ một danh sách các thư mục (được phân tách bằng dấu hai chấm) có chứa các chương trình có thể thực thi và đủ điều kiện để trở thành lệnh từ vỏ Linux.

```
$ echo $PATH
/home/user/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

Để thêm một thư mục mới vào biến, bạn sẽ cần sử dụng dấu hai chấm (:).

```
$ PATH=$PATH:new_directory
```

Sau đây là một ví dụ:

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$ PATH=$PATH:/home/user/bin
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/user/bin
```

Như bạn thấy, `$PATH` được sử dụng trong giá trị mới được gán cho `PATH`. Biến này được giải quyết trong quá trình thực thi lệnh và sẽ đảm bảo rằng nội dung ban đầu của biến được giữ nguyên. Tất nhiên, bạn cũng có thể sử dụng các biến khác trong quá trình gán:

```
$ mybin=/opt/bin
$ PATH=$PATH:$mybin
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/user/bin:/opt/bin
```

Biến `PATH` cần được xử lý một cách thận trọng vì nó rất quan trọng trong quá trình làm việc trên dòng lệnh. Hãy xem xét biến `PATH` sau:

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Để tìm hiểu cách vờ gọi một lệnh cụ thể như thế nào, bạn có thể chạy lệnh `which` với tên của lệnh làm đối số. Ví dụ, chúng ta có thể thử tìm ra nơi `nano` được lưu trữ:

```
$ which nano
/usr/bin/nano
```

Như có thể thấy, tệp thực thi `nano` nằm trong thư mục `/usr/bin`. Hãy xóa thư mục khỏi biến và kiểm tra xem lệnh có còn hoạt động hay không:

```
$ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games
$ echo $PATH
```

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games
```

Hãy tra cứu lại lệnh `nano` một lần nữa:

```
$ which nano
which: no nano in (/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games)
```

Như bạn có thể thấy, lệnh không được tìm thấy, do đó không được thực thi. Thông báo lỗi cũng giải thích lý do tại sao lệnh không được tìm thấy và nó đã được tìm kiếm ở những vị trí nào.

Hãy thêm lại các thư mục và thử chạy lại lệnh.

```
$ PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$ which nano
/usr/bin/nano
```

Bây giờ lệnh đã hoạt động trở lại.

TIP

Thứ tự của các thành phần trong `PATH` cũng xác định thứ tự tra cứu. Tập đầu tiên trùng và có thể thực thi được tìm thấy trong khi đi qua các đường dẫn sẽ được xử lý.

Bài tập Hướng dẫn

1. Hãy tạo một biến cục bộ `number`.

2. Hãy tạo một biến môi trường `ORDER`, sử dụng một trong hai phương pháp trên.

3. Hãy hiển thị cả tên biến và nội dung của chúng.

4. Phạm vi của các biến được tạo trước đó là gì?

Bài tập Mở rộng

1. Hãy tạo một biến cục bộ `nr_files` và gán số dòng tìm thấy trong tệp `/etc/passwd`. Gợi ý: Xem lệnh `wc` và phép thay thế lệnh. Đừng quên dấu trích dẫn kép.

2. Hãy tạo một biến môi trường `ME` và gán giá trị của biến `USER` cho nó.

3. Hãy nối giá trị của biến `HOME` vào `ME` có chứa dấu phân tách `:`. Hãy hiển thị nội dung của biến `ME`.

4. Sử dụng ví dụ về ngày ở trên, hãy tạo một biến có tên `today` (hôm nay) và gán ngày cho một trong các múi giờ.

5. Hãy tạo một biến khác có tên `today1` và gán ngày của hệ thống cho biến đó.

Tóm tắt

Trong bài học này, bạn đã học về:

- Các loại biến
- Cách tạo biến
- Cách thao tác với các biến

Các lệnh được dùng trong bài tập:

env

Hiển thị môi trường hiện tại.

echo

Xuất văn bản.

export

Cung cấp các biến cục bộ cho các quy trình con.

unset

Loại bỏ một biến.

Đáp án Bài tập Hướng dẫn

1. Hãy tạo một biến cục bộ `number`.

```
$ number=5
```

2. Hãy tạo một biến môi trường `ORDER`, sử dụng một trong hai phương pháp trên.

```
$ export ORDER=desc
```

3. Hãy hiển thị cả tên biến và nội dung của chúng.

```
$ echo number
number
$ echo ORDER
ORDER
$ echo $number
5
$ echo $ORDER
desc
```

4. Phạm vi của các biến được tạo trước đó là gì?

- Phạm vi của biến cục bộ `number` chỉ là vỏ hiện tại.
- Phạm vi của biến môi trường `ORDER` là vỏ hiện tại và tất cả các vỏ con được tạo từ nó.

Đáp án Bài tập Mở rộng

1. Hãy tạo một biến cục bộ `nr_files` và gán số dòng tìm thấy trong tệp `/etc/passwd`. Gợi ý: Xem lệnh `wc` và phép thay thế lệnh. Đừng quên dấu trích dẫn kép.

```
$ nr_files=`wc -l /etc/passwd`
```

2. Hãy tạo một biến môi trường `ME` và gán giá trị của biến `USER` cho nó.

```
$ export ME=$USER
```

3. Hãy nối giá trị của biến `HOME` vào `ME` có chứa dấu phân tách `:`. Hãy hiển thị nội dung của biến `ME`.

```
$ ME=$ME:$HOME
$ echo $ME
user:/home/user
```

4. Sử dụng ví dụ về ngày ở trên, hãy tạo một biến có tên `today` (hôm nay) và gán ngày cho một trong các múi giờ.

Phần sau sử dụng múi giờ GMT và EST làm ví dụ, nhưng mọi lựa chọn múi giờ đều sẽ hợp lệ.

```
$ today=$(TZ=GMT date)
$ echo $today
Thu 31 Jan 15:07:35 GMT 2019
```

hoặc

```
$ today=$(TZ=EST date)
$ echo $today
Thu 31 Jan 10:07:35 EST 2019
```

5. Hãy tạo một biến khác có tên `today1` và gán ngày của hệ thống cho biến đó.

Giả sử rằng bạn đang ở múi giờ GMT:

```
$ today1=$(date)
```

```
$ echo $today1
```

```
Thu 31 Jan 10:07:35 EST 2019
```



**Linux
Professional
Institute**

2.2 Sử dụng Dòng lệnh để Nhận Trợ giúp

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 2.2](#)

Khối lượng

2

Các lĩnh vực kiến thức chính

- Trang hướng dẫn
- Trang thông tin

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

- `man`
- `info`
- `/usr/share/doc/`
- `locate`



2.2 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	2 Tìm đường trong Hệ thống Linux
Mục tiêu:	2.2 Nhận trợ giúp trên Dòng lệnh
Bài:	1 trên 1

Giới thiệu

Dòng lệnh là một công cụ rất phức tạp. Mỗi một lệnh sẽ có các tùy chọn riêng; do vậy, tài liệu chính là chìa khóa để làm việc với hệ thống Linux. Ngoài thư mục `/usr/share/doc/` là nơi lưu trữ hầu hết các tài liệu, có nhiều công cụ khác cũng cung cấp thông tin về cách sử dụng các lệnh trong Linux. Trong bài học này, chúng ta sẽ tập trung vào các phương pháp để có thể truy cập vào các tài liệu đó và nhận được những hỗ trợ cần thiết.

Trong dòng lệnh Linux, có vô số các cách khác nhau để nhận trợ giúp. `man`, `help` và `info` là một vài ví dụ trong số đó. Đối với Linux Essentials, ta sẽ tập trung vào `man` và `info` vì chúng là những công cụ được sử dụng phổ biến nhất để có thể nhận được trợ giúp.

Một chủ đề khác thuộc bài học này là việc định vị tệp. Ta sẽ chủ yếu làm việc với lệnh `locate`.

Nhận Trợ giúp trên Dòng lệnh

Trợ giúp được tích hợp sẵn

Khi bắt đầu với tham số `--help`, hầu hết các lệnh sẽ hiển thị một số hướng dẫn ngắn gọn về cách

sử dụng lệnh. Tuy không phải lệnh nào cũng có tính năng này nhưng chúng ta vẫn nên cố gắng tìm hiểu trước về các tham số của lệnh để việc sử dụng chúng được hiệu quả hơn. Hãy lưu ý rằng các hướng dẫn từ `--help` thường khá ngắn gọn so với các nguồn tài liệu khác mà chúng ta sẽ cùng thảo luận tới đây trong bài học này.

Trang Hướng dẫn

Hầu hết các lệnh đều sẽ cung cấp một trang hướng dẫn (trang "manual"). Tài liệu này thường được cài đặt cùng với phần mềm và có thể được truy cập bằng lệnh `man`. Lệnh cần hiển thị trang hướng dẫn sẽ được thêm vào `man` làm đối số:

```
$ man mkdir
```

Lệnh này sẽ mở ra trang hướng dẫn cho `mkdir`. Bạn có thể sử dụng các phím mũi tên lên và xuống hoặc phím cách để điều hướng trong trang hướng dẫn. Để thoát khỏi trang hướng dẫn, hãy nhấn phím `q`.

Mỗi một trang hướng dẫn sẽ được chia thành tối đa 11 phần, dù đa phần trong số đó sẽ là tùy chọn:

Mục	Mô tả
NAME	Tên lệnh và mô tả ngắn gọn
SYNOPSIS	Mô tả cú pháp của lệnh
DESCRIPTION	Mô tả tác dụng của lệnh
OPTIONS	Các tùy chọn có sẵn
ARGUMENTS	Các đối số có sẵn
FILES	Các tệp phụ trợ
EXAMPLES	Một ví dụ về dòng lệnh
SEE ALSO	Tham chiếu chéo đến các chủ đề liên quan
DIAGNOSTICS	Thông báo Cảnh báo và Lỗi
COPYRIGHT	(Các) Tác giả của lệnh
BUGS	Các hạn chế đã được xác định của lệnh

Trên thực tế, hầu hết các trang hướng dẫn sẽ không có tất cả các mục này.

Các trang hướng dẫn được tổ chức thành 8 loại danh mục, được đánh số từ 1 đến 8:

Danh mục	Mô tả
1	Lệnh người dùng
2	Cuộc gọi hệ thống
3	Chức năng của thư viện C
4	Trình điều khiển và tệp thiết bị
5	Tệp cấu hình và định dạng tệp
6	Trò chơi
7	Khác
8	Lệnh quản trị hệ thống
9	Chức năng của hạt nhân (không chuẩn)

Mỗi trang hướng dẫn sẽ thuộc về đúng một danh mục duy nhất. Tuy nhiên, nhiều danh mục có thể chứa các trang hướng dẫn có cùng tên. Hãy lấy lệnh `passwd` làm ví dụ. Lệnh này có thể được sử dụng để thay đổi mật khẩu của người dùng. Vì `passwd` là một lệnh của người dùng nên trang hướng dẫn của nó nằm trong danh mục 1. Ngoài lệnh `passwd`, tệp cơ sở dữ liệu mật khẩu `/etc/passwd` cũng có một trang hướng dẫn được gọi là `passwd`. Vì tệp này là tệp cấu hình nên nó thuộc danh mục 5. Khi nhắc đến một trang hướng dẫn, danh mục thường sẽ được thêm vào tên của trang hướng dẫn, như trong `passwd(1)` hoặc `passwd(5)`, để xác định trang hướng dẫn tương ứng.

Theo mặc định, man `passwd` sẽ hiển thị trang hướng dẫn đầu tiên có sẵn, trong trường hợp này là `passwd(1)`. Danh mục của trang hướng dẫn ta cần có thể sẽ được ghi rõ trong một lệnh, chẳng hạn như `man 1 passwd` hoặc `man 5 passwd`.

Chúng ta đã thảo luận về cách điều hướng trong trang hướng dẫn và cách quay lại dòng lệnh. Trong phạm vi nội bộ, man sử dụng lệnh `less` để hiển thị nội dung của trang hướng dẫn. `less` cho phép bạn tìm kiếm văn bản trong một trang hướng dẫn. Để tìm kiếm từ `linux`, bạn chỉ cần sử dụng `/linux` để tìm kiếm tiếp từ điểm bạn đang ở trên trang, hoặc `?linux` để bắt đầu tìm kiếm ngược trở lại. Tác vụ này sẽ làm nổi bật tất cả các kết quả phù hợp và di chuyển trang đến kết quả phù hợp được đánh dấu đầu tiên. Trong cả hai trường hợp, bạn có thể gõ phím `N` để chuyển sang kết quả tìm kiếm tiếp theo. Để tìm thêm thông tin về các tính năng bổ sung này, hãy nhấn phím `H` và một menu có tất cả các thông tin sẽ xuất hiện.

Trang Thông tin

Một công cụ khác có thể trợ giúp bạn trong quá trình làm việc với hệ thống Linux là các Trang Thông tin. Các trang thông tin thường sẽ chi tiết hơn các trang hướng dẫn và sẽ được định dạng ở

dạng siêu văn bản tương tự như các trang web trên Internet.

Các trang thông tin có thể được hiển thị như thế này:

```
$ info mkdir
```

Đối với mỗi trang thông tin, lệnh `info` sẽ đọc một tệp thông tin được cấu trúc thành các nút riêng lẻ trong một cây. Mỗi nút sẽ chứa một chủ đề đơn giản và lệnh `info` chứa các siêu liên kết có thể giúp bạn di chuyển từ nút này sang nút khác. Bạn có thể truy cập liên kết bằng cách nhấn enter trong khi đặt con trỏ vào một trong các dấu hoa thị ở đầu.

Tương tự như man, công cụ `info` cũng có các lệnh điều hướng trang. Bạn có thể tìm hiểu thêm về các lệnh này bằng cách nhấn phím `?` khi đang ở trên trang thông tin. Những công cụ này sẽ giúp bạn điều hướng trang dễ dàng hơn cũng như hiểu cách truy cập các nút và di chuyển trong cây nút.

Thư mục `/usr/share/doc/`

Như đã đề cập trước đây, thư mục `/usr/share/doc/` sẽ lưu trữ hầu hết các tài liệu về các lệnh mà hệ thống đang sử dụng. Thư mục này chứa một thư mục cho hầu hết các gói được cài đặt trên hệ thống. Tên của thư mục thường là tên của gói và đôi khi là tên phiên bản của nó. Các thư mục này sẽ bao gồm tệp `README` hoặc `readme.txt` chứa tài liệu cơ bản của gói. Bên cạnh tệp `README`, thư mục cũng có thể chứa các tệp tài liệu khác, chẳng hạn như nhật ký thay đổi bao gồm chi tiết lịch sử của chương trình hoặc ví dụ về tệp cấu hình cho gói cụ thể.

Thông tin trong tệp `README` có thể sẽ thay đổi tùy theo từng gói. Vì tất cả các tệp đều được viết bằng văn bản thuần túy nên chúng có thể được đọc bằng bất kỳ trình soạn thảo văn bản nào. Số lượng và loại tệp chính xác cũng sẽ phụ thuộc vào từng gói. Hãy kiểm tra một số thư mục để có cái nhìn tổng quan về nội dung của chúng.

Định vị Tệp

Lệnh `locate`

Một hệ thống Linux được xây dựng từ nhiều thư mục và tệp. Linux có nhiều công cụ để định vị một tệp cụ thể trong hệ thống. Cách nhanh nhất là sử dụng lệnh `locate`. `locate` sẽ tìm kiếm trong cơ sở dữ liệu và sau đó xuất tất cả các tên khớp với chuỗi đã cho:

```
$ locate note
/lib/udev/keymaps/zepto-znote
```

```

/usr/bin/zipnote
/usr/share/doc/initramfs-tools/maintainer-notes.html
/usr/share/man/man1/zipnote.1.gz

```

Lệnh `locate` cũng hỗ trợ việc sử dụng các ký tự đại diện (wildcard) và biểu thức chính quy; do đó, chuỗi tìm kiếm không nhất thiết phải khớp với toàn bộ tên của tệp bạn muốn tìm. Ta sẽ tìm hiểu thêm về biểu thức chính quy trong bài học sau.

Theo mặc định, `locate` hoạt động như thể mẫu tìm kiếm (pattern) được bao quanh bởi các dấu hoa thị, vì vậy `locate PATTERN` cũng giống như `locate *PATTERN*`. Điều này cho phép bạn chỉ cần cung cấp các chuỗi con thay vì tên tệp chính xác. Hành vi này có thể được thay đổi bằng các tùy chọn khác nhau được giải thích trong trang hướng dẫn của `locate`.

Vì `locate` đang đọc từ cơ sở dữ liệu nên bạn có thể sẽ không tìm thấy tệp mà bạn đã tạo gần đây. Cơ sở dữ liệu được quản lý bởi một chương trình có tên `updatedb`. Thông thường thì chương trình này được chạy định kỳ; nhưng nếu bạn có quyền gốc và cần cập nhật cơ sở dữ liệu ngay lập tức, bạn có thể tự mình chạy lệnh `updatedb` bất kỳ lúc nào.

Lệnh find

`find` là một công cụ phổ biến khác được sử dụng để tìm kiếm tệp. Lệnh này có cách tiếp cận khác so với lệnh `locate`. Lệnh `find` tìm kiếm đệ quy (tìm kiếm tự gọi) cây thư mục, bao gồm cả các thư mục con của nó. `find` thực hiện tác vụ tìm kiếm như vậy ở mỗi lần gọi và không quản lý một cơ sở dữ liệu như `locate`. Tương tự như `locate`, `find` cũng hỗ trợ các ký tự đại diện và biểu thức chính quy. `find` yêu cầu phải có ít nhất một đường dẫn để tìm kiếm.. Hơn nữa, cái gọi là biểu thức có thể được thêm vào để cung cấp tiêu chí lọc cho tệp nào sẽ hiển thị. Một ví dụ là biểu thức `-name` sẽ tìm kiếm các tệp có một cái tên cụ thể:

```

~$ cd Downloads
~/Downloads
$ find . -name thesis.pdf
./thesis.pdf
~/Downloads
$ find ~ -name thesis.pdf
/home/carol/Downloads/thesis.pdf

```

Lệnh `find` đầu tiên sẽ tìm kiếm tệp trong thư mục `Downloads` hiện tại, trong khi lệnh thứ hai sẽ tìm kiếm tệp trong thư mục chính của người dùng.

Lệnh `find` rất phức tạp, do đó nó sẽ không được đề cập đến trong bài kiểm tra Linux Essentials. Tuy nhiên, nó là một công cụ rất mạnh mẽ và đặc biệt hữu ích trong thực tế.

Bài tập Hướng dẫn

1. Hãy sử dụng lệnh man để tìm hiểu chức năng của từng lệnh:

Lệnh	Mô tả
ls	Hiển thị nội dung của một thư mục.
cat	
cut	
cd	
cp	
mv	
mkdir	
touch	
wc	
passwd	
rm	
rmdir	
more	
less	
whereis	
head	
tail	
sort	
tr	
chmod	
grep	

2. Hãy mở trang thông tin ls và xác định MENU.

- Bạn có những lựa chọn nào?

- Hãy tìm ra tùy chọn cho phép bạn sắp xếp đầu ra theo thời gian sửa đổi.

3. Hãy hiển thị đường dẫn đến 3 tệp README đầu tiên. Hãy sử dụng lệnh man để xác định tùy chọn chính xác cho locate.

4. Hãy tạo một tệp có tên test trong thư mục chính của bạn. Hãy tìm đường dẫn tuyệt đối của nó bằng lệnh locate.

5. Bạn có ngay lập tức tìm thấy nó không? Bạn phải làm gì để locate có thể tìm thấy nó?

6. Hãy tìm kiếm tệp test mà bạn đã tạo trước đó bằng cách sử dụng lệnh find. Bạn đã sử dụng cú pháp nào và đường dẫn tuyệt đối là gì?

Bài tập Mở rộng

1. Có một lệnh trong bảng trên không có trang hướng dẫn. Đó là lệnh nào và theo bạn thì tại sao nó lại không có trang hướng dẫn?

2. Sử dụng các lệnh trong bảng trên, hãy tạo cây tệp sau. Các tên bắt đầu bằng chữ hoa là Thư mục và tên viết thường là Tệp.

```
User
├── Documents
│   ├── Hello
│   │   ├── hey2
│   │   ├── helloa
│   │   └── ola5
│   └── World
│       └── earth9
├── Downloads
│   ├── Music
│   └── Songs
│       ├── collection1
│       └── collection2
├── Test
│   └── passa
└── test
```

3. Hãy hiển thị lên màn hình thư mục làm việc hiện tại, bao gồm cả các thư mục con.

4. Hãy tìm kiếm trong cây tất cả các tệp kết thúc bằng một chữ số.

5. Hãy xoá toàn bộ cây thư mục bằng một lệnh duy nhất.

Tóm tắt

Trong bài học này, bạn đã học về:

- Cách để nhận được trợ giúp
- Cách sử dụng lệnh `man`
- Cách điều hướng trang hướng dẫn
- Các phần khác nhau của trang hướng dẫn
- Cách sử dụng lệnh `info`
- Cách điều hướng giữa các nút khác nhau
- Cách tìm kiếm các tệp trong hệ thống

Các lệnh dùng trong bài tập:

man

Hiển thị một trang hướng dẫn.

info

Hiển thị một trang thông tin.

locate

Tìm kiếm các tệp có tên cụ thể trong cơ sở dữ liệu `locate`.

find

Tìm kiếm trong hệ thống tệp các tên phù hợp với một bộ tiêu chí lựa chọn.

updatedb

Cập nhật cơ sở dữ liệu `locate`.

Đáp án Bài tập Hướng dẫn

1. Hãy sử dụng lệnh man để tìm hiểu chức năng của từng lệnh:

Lệnh	Mô tả
ls	Hiển thị nội dung của một thư mục
cat	Ghép nối hoặc xem các tệp văn bản
cut	Xóa các phần khỏi tệp văn bản
cd	Thay đổi sang một thư mục khác
cp	Sao chép một tệp
mv	Di chuyển một tệp hoặc đổi tên
mkdir	Tạo một thư mục mới
touch	Tạo một tệp hoặc sửa đổi ngày và giờ sửa đổi lần cuối của một tệp hiện có
wc	Đếm số từ, dòng hoặc byte của một tệp
passwd	Thay đổi mật khẩu của người dùng
rm	Xóa một tệp
rmdir	Xóa một thư mục
more	Xem các tệp văn bản trên từng màn hình một
less	Xem các tệp văn bản, cho phép cuộn lên và xuống từng dòng hoặc từng trang một
whereis	Hiển thị đường dẫn tệp đến một chương trình được chỉ định và các tệp hướng dẫn có liên quan
head	Hiển thị một vài dòng đầu tiên của một tệp
tail	Hiển thị vài dòng cuối cùng của một tệp
sort	Sắp xếp một tệp theo thứ tự số hoặc bảng chữ cái
tr	Dịch hoặc xóa các ký tự của tệp
chmod	Thay đổi quyền của tệp
grep	Tìm kiếm trong một tệp

2. Hãy mở trang thông tin `ls` và xác định MENU.

- Bạn có những lựa chọn nào?
 - Tập nào được liệt kê
 - Thông tin gì được liệt kê
 - Sắp xếp đầu ra
 - Chi tiết về sắp xếp phiên bản
 - Định dạng đầu ra chung
 - Định dạng dấu thời gian của tập
 - Định dạng tên tập
- Hãy tìm ra tùy chọn cho phép bạn sắp xếp đầu ra theo thời gian sửa đổi.

```
-t hoặc --sort=time
```

3. Hãy hiển thị đường dẫn đến 3 tập `README` đầu tiên. Hãy sử dụng lệnh `man` để xác định tùy chọn chính xác cho `locate`.

```
$ locate -l 3 README
/etc/alternatives/README
/etc/init.d/README
/etc/rc0.d/README
```

4. Hãy tạo một tập có tên `test` trong thư mục chính của bạn. Hãy tìm đường dẫn tuyệt đối của nó bằng lệnh `locate`.

```
$ touch test
$ locate test
/home/user/test
```

5. Bạn có ngay lập tức tìm thấy nó không? Bạn phải làm gì để `locate` có thể tìm thấy nó?

```
$ sudo updatedb
```

Tập vừa mới được tạo, do đó nó không được ghi lại trong cơ sở dữ liệu.

6. Hãy tìm kiếm tập `test` mà bạn đã tạo trước đó bằng cách sử dụng lệnh `find`. Bạn đã sử dụng cú pháp nào và đường dẫn tuyệt đối là gì?

```
$ find ~ -name test
```

hoặc

```
$ find . -name test  
/home/user/test
```

Đáp án Bài tập Mở rộng

1. Có một lệnh trong bảng trên không có trang hướng dẫn. Đó là lệnh nào và theo bạn thì tại sao nó lại không có trang hướng dẫn?

Lệnh `cd`. Nó không có trang hướng dẫn vì nó là lệnh vô tích hợp sẵn.

2. Sử dụng các lệnh trong bảng trên, hãy tạo cây tệp sau. Các tên bắt đầu bằng chữ hoa là Thư mục và tên viết thường là Tệp.

```
User
├── Documents
│   ├── Hello
│   │   ├── hey2
│   │   ├── helloa
│   │   └── ola5
│   └── World
│       └── earth9
├── Downloads
│   ├── Music
│   └── Songs
│       ├── collection1
│       └── collection2
├── Test
│   └── passa
└── test
```

Giải pháp ở đây là sự kết hợp của các lệnh `mkdir` và `touch`.

3. Hãy hiển thị lên màn hình thư mục làm việc hiện tại, bao gồm cả các thư mục con.

```
$ ls -R
```

4. Hãy tìm kiếm trong cây tất cả các tệp kết thúc bằng một chữ số.

```
$ find ~ -name "[0-9]"
$ locate "[0-9]"
```

5. Hãy xoá toàn bộ cây thư mục bằng một lệnh duy nhất.

```
$ rm -r Documents Downloads Test test
```



2.3 Sử dụng Thư mục và Tập liệt kê

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 2.3](#)

Khối lượng

2

Các lĩnh vực kiến thức chính

- Tập, thư mục
- Tập và thư mục ẩn
- Thư mục chính
- Đường dẫn tuyệt đối và tương đối

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

- Các tùy chọn phổ biến cho `ls`
- Danh sách đệ quy
- `cd`
- `.` và `..`
- `home` và `~`



2.3 Bài 1

Giới thiệu

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	2 Tìm đường trong Hệ thống Linux
Mục tiêu:	2.3 Sử dụng Thư mục và Tập Liệt kê
Bài:	1 trên 2

Tệp và Thư mục

Hệ thống tệp của Linux cũng giống như các hệ thống tệp của các hệ điều hành khác ở chỗ nó cũng chứa các *tệp* và *thư mục*. Các tệp thường chứa dữ liệu như văn bản mà con người có thể đọc được, các chương trình thực thi được hoặc dữ liệu nhị phân được máy tính sử dụng. Các thư mục được sử dụng để tổ chức bên trong hệ thống tệp. Các thư mục có thể chứa các tệp và thư mục khác.

```
$ tree
```

```
Documents
├── Mission-Statement.txt
└── Reports
    └── report2018.txt
```

```
1 directory, 2 files
```

Trong ví dụ này, Documents là một thư mục chứa một tệp (`Mission-Statement.txt`) và một *thư mục con* (Reports). Thư mục Reports cũng chứa một tệp có tên `report2018.txt`. Thư mục Documents được coi là *thư mục mẹ* của thư mục Reports.

TIP Nếu lệnh `tree` không có sẵn trên hệ thống của bạn, hãy cài đặt nó bằng trình quản lý gói của bản phân phối Linux của bạn. Tham khảo bài học về trình quản lý gói để biết cách thực hiện.

Tên Tệp và Thư mục

Tên tệp và thư mục trong Linux có thể chứa chữ thường, chữ hoa, số, dấu cách và ký tự đặc biệt. Tuy nhiên, vì nhiều ký tự đặc biệt có ý nghĩa đặc biệt trong vỏ Linux nên tốt nhất là không sử dụng dấu cách hoặc ký tự đặc biệt khi đặt tên tệp hoặc thư mục. Ví dụ, phím cách cần phải có thêm *ký tự thoát* `\` để được nhập chính xác:

```
$ cd Mission\ Statements
```

Ngoài ra, hãy xem tên tệp `report2018.txt`. Tên tệp có thể chứa một *hậu tố* đứng sau dấu chấm (`.`). Không giống như Windows, hậu tố này không có ý nghĩa đặc biệt trong Linux; nó ở đó chỉ để giúp con người đọc hiểu. Trong ví dụ của chúng ta, `.txt` cho ta biết rằng đây là tệp văn bản gốc (mặc dù về mặt kỹ thuật thì nó có thể chứa bất kỳ loại dữ liệu nào).

Điều hướng Hệ thống tệp

Nhận Vị trí Hiện tại

Vì các vỏ Linux như Bash đều dựa trên văn bản nên điều quan trọng là phải nhớ vị trí hiện tại của bạn khi điều hướng hệ thống tệp. *Dấu nhắc Lệnh* sẽ cung cấp thông tin này:

```
user@hostname ~/Documents/Reports $
```

Hãy lưu ý rằng các thông tin như `user` (người dùng) và `hostname` (tên máy chủ) sẽ được đề cập trong các phần sau. Từ dấu nhắc lệnh, bây giờ chúng ta đã biết được rằng vị trí hiện tại của ta là ở trong thư mục Reports. Tương tự, lệnh `pwd` sẽ *in đường dẫn của thư mục làm việc*:

```
user@hostname ~/Documents/Reports $ pwd
/home/user/Documents/Reports
```


Mối quan hệ của các thư mục được thể hiện bằng dấu gạch chéo (/). Ta biết rằng Reports là thư mục con của Documents; Documents là thư mục con của user nằm trong thư mục có tên home. home trông có vẻ như không có thư mục mẹ, nhưng điều đó hoàn toàn không đúng. Thư mục mẹ của home được gọi là thư mục gốc (root) và được biểu thị bằng dấu gạch chéo đầu tiên (/). Chúng ta sẽ thảo luận về thư mục gốc trong phần sau.

Hãy lưu ý rằng đầu ra của lệnh pwd sẽ hơi khác so với đường dẫn được cung cấp trên dấu nhắc lệnh. Thay vì /home/user, dấu nhắc lệnh sẽ chứa dấu ngã (~). Dấu ngã là một ký tự đặc biệt đại diện cho thư mục chính của người dùng. Điều này sẽ được đề cập chi tiết hơn trong bài học tiếp theo.

Liệt kê Nội dung Thư mục

Nội dung của thư mục hiện tại được liệt kê bằng lệnh ls:

```
user@hostname ~/Documents/Reports $ ls
report2018.txt
```

Hãy lưu ý rằng ls sẽ không cung cấp thông tin về thư mục mẹ. Tương tự, theo mặc định, ls sẽ không hiển thị bất kỳ thông tin nào về nội dung của các thư mục con. ls chỉ có thể “xem” những nội dung nào có ở trong thư mục hiện tại.

Thay đổi Thư mục hiện tại

Điều hướng trong Linux chủ yếu được thực hiện bằng lệnh cd (changes directory). Lệnh này sẽ *thay đổi thư mục hiện tại*. Bằng cách sử dụng lệnh pwd như đã biết, chúng ta có thể thấy thư mục hiện tại của mình là /home/user/Documents/Reports. Ta có thể thay đổi thư mục hiện tại của mình bằng cách nhập một đường dẫn mới:

```
user@hostname ~ $ cd /home/user/Documents
user@hostname ~/Documents $ pwd
/home/user/Documents
user@hostname ~/Documents $ ls
Mission-Statement.txt Reports
```

Từ vị trí mới này, ta có thể “xem” Mission-Statement.txt và thư mục con Reports, nhưng không thể xem được nội dung của thư mục con. Chúng ta có thể điều hướng trở lại Reports bằng cách như sau:

```
user@hostname ~/Documents $ cd Reports
```

```
user@hostname ~/Documents/Reports $ pwd
/home/user/Documents/Reports
user@hostname ~/Documents/Reports $ ls
report2018.txt
```

Bây giờ ta đã trở lại vị trí ban đầu.

Đường dẫn Tuyệt đối và Tương đối

Lệnh `pwd` luôn in một *đường dẫn tuyệt đối*. Điều này có nghĩa là đường dẫn này chứa mọi bước của đường dẫn, từ đầu (`/`) đến cuối (Reports) hệ thống tệp. Đường dẫn tuyệt đối sẽ luôn bắt đầu bằng dấu `/`.

```
/
├── home
│   ├── user
│   │   ├── Documents
│   │   └── Reports
```

Đường dẫn tuyệt đối sẽ chứa tất cả những thông tin cần thiết để truy cập Reports từ bất kỳ đâu trong hệ thống tệp. Hạn chế của nó là phải gõ toàn bộ đường dẫn bằng tay.

Ví dụ thứ hai là (`cd Reports`) thì dễ gõ hơn nhiều. Đây là một ví dụ về *đường dẫn tương đối*. Đường dẫn tương đối sẽ ngắn hơn và ý nghĩa của nó chỉ liên quan đến vị trí hiện tại của bạn. Hãy cùng loại suy như thế này: Tôi đến chơi với bạn tại nhà của bạn. Bạn nói với tôi rằng bạn của bạn sống ở ngay bên cạnh. Tôi sẽ hiểu ngay vị trí của nhà người bạn đó là ở đâu vì nó có liên quan đến vị trí hiện tại của tôi. Nhưng nếu bạn nói với tôi điều này qua điện thoại, tôi sẽ không thể tìm thấy nhà của người bạn kia. Bạn sẽ phải cung cấp cho tôi một địa chỉ bao gồm thông tin đường phố đầy đủ.

Đường dẫn Tương đối Đặc biệt

Vỏ Linux cung cấp cho chúng ta các cách để rút ngắn đường dẫn khi điều hướng. Để đưa ra các đường dẫn đặc biệt đầu tiên, ta sẽ nhập lệnh `ls` với cờ `-a`. Cờ này sẽ sửa đổi lệnh `ls` để *tất cả* các tệp và thư mục được liệt kê, bao gồm cả các tệp và thư mục ẩn:

```
user@hostname ~/Documents/Reports $ ls -a
.
..
report2018.txt
```

NOTE | Bạn có thể tham khảo trang hướng dẫn cho `ls` để hiểu `-a` đang làm gì ở đây.

Lệnh này đã cho ta thấy hai kết quả bổ sung: Đây là những đường dẫn đặc biệt. Chúng không đại diện cho các tệp hoặc thư mục mới, mà chúng đại diện cho các thư mục mà bạn đã biết:

.

Cho biết vị trí hiện tại (trong trường hợp này là Reports).

..

Cho biết thư mục mẹ (trong trường hợp này là Documents).

Thông thường, ta không cần phải sử dụng đường dẫn tương đối đặc biệt cho vị trí hiện tại. Việc nhập `report2018.txt` sẽ dễ dàng và dễ hiểu hơn so với việc nhập `./report2018.txt`. Nhưng . có những cách sử dụng mà bạn có thể tìm hiểu trong các phần sau. Hiện tại, ta sẽ tập trung vào đường dẫn tương đối cho thư mục mẹ:

```
user@hostname ~/Documents/Reports $ cd ..
user@hostname ~/Documents $ pwd
/home/user/Documents
```

Ví dụ về `cd` sẽ dễ dàng hơn nhiều khi sử dụng `..` thay vì đường dẫn tuyệt đối. Ngoài ra, chúng ta có thể kết hợp mẫu này để điều hướng cây tệp rất nhanh.

```
user@hostname ~/Documents $ cd ../../
$ pwd
/home
```

Bài tập Hướng dẫn

- Đối với mỗi đường dẫn sau, hãy xác định xem đó là đường dẫn *tuyệt đối* hay *tương đối*:

/home/user/Downloads	
../Reports	
/var	
docs	
/	

- Hãy quan sát cấu trúc tệp sau. Lưu ý: Các thư mục sẽ kết thúc bằng dấu gạch chéo (/) khi `tree` được gọi với tùy chọn `-F`. Bạn sẽ phải có các đặc quyền nâng cao để chạy lệnh `tree` trên thư mục gốc (/). Sau đây là đầu ra ví dụ và nó không ám chỉ cấu trúc thư mục đầy đủ. Hãy sử dụng nó để trả lời các câu hỏi sau:

```
$ sudo tree -F /
```

```
/
├── etc/
│   ├── network/
│   │   └── interfaces
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   └── system.conf
│   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d/
│       └── udev.conf
└── home/
    ├── lost+found/
    └── user/
        └── Documents/
```

```
12 directories, 5 files
```

Hãy sử dụng cấu trúc này để trả lời các câu hỏi sau.

Người dùng nhập các lệnh sau:

```
$ cd /etc/udev
$ ls -a
```

Đầu ra của lệnh `ls -a` sẽ là gì?

3. Hãy nhập lệnh ngắn nhất có thể cho mỗi lệnh sau:

- Vị trí hiện tại của bạn là root (/). Hãy nhập lệnh để điều hướng đến `lost+found` trong thư mục `home` (ví dụ):

```
$ cd home/lost+found
```

- Vị trí hiện tại của bạn là root (/). Hãy nhập lệnh để điều hướng đến thư mục có tên `/etc/network/`.
-

- Vị trí hiện tại của bạn là `/home/user/Documents/`. Hãy điều hướng đến thư mục có tên `/etc/`.
-

- Vị trí hiện tại của bạn là `/etc/systemd/system/`. Hãy điều hướng đến thư mục có tên `/home/user/`.
-

4. Hãy xem xét các lệnh sau:

```
$ pwd
/etc/udev/rules.d
$ cd ../../systemd/user
$ cd ..
$ pwd
```

Đầu ra của lệnh `pwd` cuối cùng là gì?

Bài tập Mở rộng

1. Giả sử người dùng đã nhập các lệnh sau:

```
$ mkdir "this is a test"
$ ls
this is a test
```

Lệnh `cd` nào sẽ cho phép bạn vào thư mục này?

2. Hãy thực hiện lại lần nữa, nhưng sau khi nhập `cd this`, hãy nhấn phím TAB. Lúc này điều gì đang được hiển thị trên dấu nhắc lệnh?

Đây là một ví dụ về tính năng *tự động hoàn thành*; đây là một công cụ vô giá không chỉ giúp tiết kiệm thời gian mà còn ngăn ngừa lỗi chính tả.

3. Hãy thử tạo một thư mục có tên chứa ký tự `\`. Hãy hiển thị tên thư mục bằng `ls` và xóa thư mục.

Tóm tắt

Trong bài học này, bạn đã học về:

- Nguyên tắc cơ bản của hệ thống tệp Linux
- Sự khác biệt giữa thư mục *mẹ* và thư mục *con*
- Sự khác biệt giữa đường dẫn tệp *tuyệt đối* và đường dẫn tệp *tương đối*
- Các đường dẫn tương đối đặc biệt `.` và `..`
- Điều hướng hệ thống tệp bằng cách sử dụng `cd`
- Hiển thị vị trí hiện tại của bạn bằng `pwd`
- Liệt kê *tất cả* các tệp và thư mục bằng cách sử dụng `ls -a`

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

`cd`

Thay đổi thư mục hiện tại.

`pwd`

In đường dẫn của thư mục làm việc hiện tại

`ls`

Liệt kê nội dung của một thư mục và hiển thị thuộc tính của các tệp

`mkdir`

Tạo một thư mục mới

`tree`

Hiển thị danh sách phân cấp của cây thư mục

Đáp án Bài tập Hướng dẫn

1. Đối với mỗi đường dẫn sau, hãy xác định xem đó là đường dẫn *tuyệt đối* hay *tương đối*:

/home/user/Downloads	tuyệt đối
../Reports	tương đối
/var	tuyệt đối
docs	tương đối
/	tuyệt đối

2. Hãy quan sát cấu trúc tệp sau. Lưu ý: Các thư mục sẽ kết thúc bằng dấu gạch chéo (/) khi `tree` được gọi với tùy chọn `-F`. Bạn sẽ phải có các đặc quyền nâng cao để chạy lệnh `tree` trên thư mục gốc (/). Sau đây là đầu ra ví dụ và nó không ám chỉ cấu trúc thư mục đầy đủ. Hãy sử dụng nó để trả lời các câu hỏi sau:

```
$ sudo tree -F /
```

```
/
├── etc/
│   ├── network/
│   │   └── interfaces
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   └── system.conf
│   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d/
│       └── udev.conf
└── home/
    ├── lost+found/
    └── user/
        └── Documents/
```

```
12 directories, 5 files
```

Người dùng nhập các lệnh sau:


```
$ cd /etc/udev
$ ls -a
```

Đầu ra của lệnh `ls -a` sẽ là gì?

```
. .. rules.d udev.conf
```

3. Hãy nhập lệnh ngắn nhất có thể cho mỗi lệnh sau:

- Vị trí hiện tại của bạn là root (/). Hãy nhập lệnh để điều hướng đến `lost+found` trong thư mục `home` (ví dụ):

```
$ cd home/lost+found
```

- Vị trí hiện tại của bạn là root (/). Hãy nhập lệnh để điều hướng đến thư mục có tên `/etc/network/`.

```
$ cd etc/network
```

- Vị trí hiện tại của bạn là `/home/user/Documents/`. Hãy điều hướng đến thư mục có tên `/etc/`.

```
$ cd /etc
```

- Vị trí hiện tại của bạn là `/etc/systemd/system/`. Hãy điều hướng đến thư mục có tên `/home/user/`

```
$ cd /home/user
```

4. Hãy xem xét các lệnh sau:

```
$ pwd
/etc/udev/rules.d
$ cd ../../systemd/user
$ cd ..
$ pwd
```

Đầu ra của lệnh `pwd` cuối cùng là gì?

```
/etc/systemd
```

Đáp án Bài tập Mở rộng

1. Giả sử người dùng đã nhập các lệnh sau:

```
$ mkdir "this is a test"
$ ls
this is a test
```

Lệnh `cd` nào sẽ cho phép bạn vào thư mục này?

```
$ cd this\ is\ a\ test
```

2. Hãy thực hiện lại lần nữa, nhưng sau khi nhập `cd this`, hãy nhấn phím TAB. Lúc này điều gì đang được hiển thị trên dấu nhắc lệnh?

```
$ cd this\ is\ a\ test
```

Đây là một ví dụ về tính năng tự động hoàn thành; đây là một công cụ vô giá không chỉ giúp tiết kiệm thời gian mà còn ngăn ngừa lỗi chính tả.

3. Hãy thử tạo một thư mục có tên chứa ký tự `\`. Hãy hiển thị tên thư mục bằng `ls` và xóa thư mục.

Bạn có thể thoát dấu gạch chéo ngược bằng một dấu gạch chéo ngược khác (`\\`), hoặc sử dụng dấu trích dẫn đơn hoặc kép xung quanh toàn bộ tên thư mục:

```
$ mkdir my\\dir
$ ls
'my\dir'
$ rmdir 'my\dir'
```



2.3 Bài 2

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	2 Tìm đường trong Hệ thống Linux
Mục tiêu:	2.3 Sử dụng Thư mục và Tập Liệt kê
Bài:	2 trên 2

Giới thiệu

Hệ điều hành Unix ban đầu được thiết kế cho các máy tính hệ thống lớn vào giữa những năm 1960. Những máy tính này được chia sẻ giữa nhiều người dùng, những người này truy cập vào tài nguyên của hệ thống thông qua *cửa sổ dòng lệnh*. Những ý tưởng cơ bản này vẫn được áp dụng xuyên suốt đến các hệ thống Linux ngày nay. Giờ đây, chúng ta vẫn nói về việc sử dụng “cửa sổ dòng lệnh” để nhập lệnh trong vỏ như thế nào, và mọi hệ thống Linux đều được tổ chức ra sao để dễ dàng tạo được nhiều người dùng trên cùng một hệ thống.

Thư mục Chính

Sau đây là một ví dụ về hệ thống tệp thông thường trong Linux:

```
$ tree -L 1 /  
/  
├─ bin  
├─ boot  
├─ cdrom  
├─ dev
```

```

├─ etc
├─ home
├─ lib
├─ mnt
├─ opt
├─ proc
├─ root
├─ run
├─ sbin
├─ srv
├─ sys
├─ tmp
├─ usr
└─ var

```

Hầu hết các thư mục này đều sẽ nhất quán trên tất cả các hệ thống Linux. Từ máy chủ đến siêu máy tính cho đến hệ thống nhúng nhỏ, người dùng Linux dày dạn kinh nghiệm có thể tự tin rằng họ có thể tìm thấy lệnh `ls` bên trong `/bin`, có thể thay đổi cấu hình hệ thống bằng cách sửa đổi tệp trong `/etc` và đọc nhật ký hệ thống trong `/var`. Vị trí tiêu chuẩn của các tệp và thư mục này được xác định bởi Tiêu chuẩn Phân cấp Hệ thống Tệp (FHS - Filesystem Hierarchy Standard) sẽ được thảo luận trong bài học sau. Ta sẽ tìm hiểu thêm về nội dung của các thư mục này trong quá trình học về Linux; nhưng hiện tại, ta đã biết rằng:

- những thay đổi mà bạn thực hiện trong hệ thống tệp gốc sẽ ảnh hưởng đến tất cả người dùng;
- việc thay đổi tệp trong hệ thống tệp gốc sẽ yêu cầu quyền của quản trị viên. Điều này có nghĩa là người dùng bình thường sẽ bị cấm sửa đổi các tệp này và thậm chí có thể bị cấm đọc các tệp này. Ta sẽ đề cập đến chủ đề về quyền trong phần sau. Bây giờ, chúng ta sẽ tập trung vào thư mục mà giờ đây đã trở nên khá quen thuộc là `/home`:

```

$ tree -L 1 /home
/home
├─ user
├─ michael
└─ lara

```

Hệ thống ví dụ của chúng ta có ba người dùng bình thường và mỗi người dùng đều có vị trí dành riêng cho họ; đó nơi họ có thể tạo và sửa đổi các tệp và thư mục mà không ảnh hưởng đến hàng xóm của mình. Ví dụ, trong bài học trước, chúng ta đã làm việc với cấu trúc tệp sau:

```

$ tree /home/user
user

```

```
└─ Documents
   └─ Mission-Statement
      └─ Reports
         └─ report2018.txt
```

Trên thực tế, hệ thống tệp có thể sẽ trông như thế này:

```
$ tree /home
/home
├─ user
│   └─ Documents
│       └─ Mission-Statement
│           └─ Reports
│               └─ report2018.txt
├─ michael
│   └─ Documents
│       └─ presentation-for-clients.odp
└─ Music
```

...và tiếp tục tiếp diễn như vậy với `lara`.

Trong Linux, `/home` cũng tương tự như một tòa nhà chung cư. Nhiều người dùng có thể có không gian riêng của họ ở đây, được chia thành các căn hộ chuyên dụng. Các tiện ích và việc bảo trì tòa nhà là trách nhiệm của người dùng gốc, tức chính chủ sở hữu của tòa nhà.

Đường dẫn Tương đối Đặc biệt cho Trang chủ chủ

Khi bạn bắt đầu một phiên mới trong cửa sổ dòng lệnh của Linux, bạn sẽ thấy một dấu nhắc lệnh tương tự như sau:

```
user@hostname ~ $
```

Dấu ngã (~) ở đây đại diện cho *thư mục chính* của chúng ta. Nếu chạy lệnh `ls`, bạn sẽ thấy một số kết quả quen thuộc:

```
$ cd ~
$ ls
Documents
```

Hãy so sánh điều này với hệ thống tệp ở trên để kiểm tra kiến thức của bạn.

Bây giờ hãy cùng xem xét những gì chúng ta đã biết về Linux: nó tương tự như một tòa nhà chung cư với nhiều người dùng cư trú tại /home. Vì vậy, nhà của user sẽ khác với nhà của người dùng michael. Để chứng minh cho điều này, chúng ta sẽ sử dụng lệnh su để *đổi người dùng*.

```
user@hostname ~ $ pwd
/home/user
user@hostname ~ $ su - michael
Password:
michael@hostname ~ $ pwd
/home/michael
```

Ý nghĩa của ~ sẽ thay đổi tùy thuộc vào người dùng là ai. Đối với michael, đường dẫn tuyệt đối của ~ là /home/michael. Đối với lara, đường dẫn tuyệt đối của ~ là /home/lara, v.v.

Đường dẫn tệp từ Tương-đối-đến-Trang-chủ

Việc sử dụng ~ cho các lệnh là rất tiện dụng, miễn là bạn không chuyển đổi người dùng. Chúng ta sẽ cùng xem xét ví dụ sau cho user khi người dùng này bắt đầu một phiên làm việc mới:

```
$ ls
Documents
$ cd Documents
$ ls
Mission-Statement
Reports
$ cd Reports
$ ls
report2018.txt
$ cd ~
$ ls
Documents
```

Hãy lưu ý rằng người dùng sẽ luôn bắt đầu một phiên mới trong thư mục chính của họ. Trong ví dụ này, user đã đi vào thư mục con Documents/Reports của họ và với lệnh cd ~, họ đã quay lại vị trí ban đầu. Bạn có thể thực hiện thao tác tương tự bằng cách sử dụng lệnh cd mà không cần đối số:

```
$ cd Documents/Reports
$ pwd
/home/user/Documents/Reports
```

```
$ cd
$ pwd
/home/user
```

Một điều cuối cùng cần lưu ý: chúng ta có thể chỉ định thư mục chính của *những người dùng khác* bằng cách chỉ định tên người dùng sau dấu ngã. Ví dụ:

```
$ ls ~michael
Documents
Music
```

Hãy lưu ý rằng điều này sẽ chỉ có tác dụng nếu `michael` đã cho phép chúng ta xem nội dung trong thư mục chính của anh ấy.

Hãy cùng xem xét tình huống mà trong đó `michael` muốn xem tệp `report2018.txt` trong thư mục chính của `user`. Giả sử rằng `michael` được cấp quyền để làm như vậy, anh ấy có thể sử dụng lệnh `less`.

```
$ less ~user/Documents/Reports/report2018.txt
```

Bất kỳ đường dẫn tệp nào chứa ký tự `~` đều được gọi là đường dẫn *tương đối đến trang chủ*.

Tệp và Thư mục ẩn

Trong bài học trước, ta đã giới thiệu tùy chọn `-a` cho lệnh `ls`. Ta đã sử dụng `ls -a` để giới thiệu hai đường dẫn tương đối đặc biệt: `.` và `..`. Tùy chọn `-a` sẽ liệt kê *tất cả* các tệp và thư mục, bao gồm cả các tệp và thư mục *ẩn*.

```
$ ls -a ~
.
..
.bash_history
.bash_logout
.bash-profile
.bashrc
Documents
```

Các tệp và thư mục ẩn sẽ luôn bắt đầu bằng dấu chấm (`.`). Theo mặc định, thư mục chính của người dùng sẽ bao gồm nhiều tệp ẩn. Chúng thường được sử dụng để đặt cấu hình dành

riêng cho người dùng và chỉ người dùng có kinh nghiệm mới sửa đổi được.

Tùy chọn Danh sách Dài

Lệnh `ls` có nhiều tùy chọn để thay đổi hành vi của nó. Hãy cùng xem một trong những lựa chọn phổ biến nhất:

```
$ ls -l
-rw-r--r-- 1 user staff      3606 Jan 13  2017 report2018.txt
```

`-l` tạo một *danh sách dài*. Mỗi tệp và thư mục sẽ chiếm một dòng, nhưng thông tin bổ sung về từng tệp và thư mục cũng sẽ được hiển thị.

`-rw-r--r--`

Loại tệp và quyền của tệp. Hãy lưu ý rằng một tệp thông thường sẽ bắt đầu bằng dấu gạch ngang và một thư mục sẽ bắt đầu bằng `d`.

`1`

Số lượng liên kết đến tệp.

`user staff`

Chỉ định quyền sở hữu của tệp. `user` là chủ sở hữu của tệp và tệp cũng được liên kết với nhóm `staff`.

`3606`

Kích thước của tệp tính bằng byte.

`Jan 13 2017`

Dấu thời gian của lần sửa đổi tệp cuối cùng.

`report2018.txt`

Tên của tệp.

Các vấn đề về quyền sở hữu, quyền hạn truy cập và liên kết sẽ được đề cập tới sau này. Như có thể thấy, phiên bản danh sách dài của lệnh `ls` đôi khi sẽ tối ưu hơn so với mặc định.

Các Tùy chọn `ls` Bổ sung

Dưới đây là một số cách mà chúng ta thường sử dụng lệnh `ls` nhất. Như có thể thấy, người dùng có thể kết hợp nhiều tùy chọn với nhau để có được đầu ra mong muốn.

ls -lh

Kết hợp *danh sách dài* với kích thước tệp *con người có thể đọc được* sẽ cung cấp cho chúng ta các hậu tố hữu ích, chẳng hạn như M cho megabyte hoặc K cho kilobyte.

ls -d */

Tùy chọn `-d` sẽ liệt kê các thư mục nhưng không liệt kê nội dung của chúng. Kết hợp nó với `*/` sẽ chỉ hiển thị các thư mục con mà không có tệp.

ls -lt

Kết hợp *danh sách dài* với tùy chọn sắp xếp theo *thời gian sửa đổi*. Các tệp có thay đổi gần đây nhất sẽ ở trên cùng và các tệp có thay đổi cũ nhất sẽ ở dưới cùng. Nhưng thứ tự này có thể được đảo ngược với:

ls -lrt

Kết hợp *danh sách dài* với *sắp xếp theo thời gian (sửa đổi)*, kết hợp với `-r` sẽ *đảo ngược* thứ tự sắp xếp. Bây giờ các tệp có thay đổi gần đây nhất sẽ nằm ở cuối danh sách. Ngoài việc sắp xếp theo thời gian *sửa đổi*, các tệp cũng có thể được sắp xếp theo thời gian *truy cập* hoặc theo thời gian *trạng thái*.

ls -lX

Kết hợp *danh sách dài* với tùy chọn sắp xếp theo *phần mở rộng của tệp*. Ví dụ: điều này sẽ nhóm tất cả các tệp kết thúc bằng `.txt` hoặc bằng `.jpg` lại với nhau, v.v.

ls -S

`-S` sẽ sắp xếp theo *kích thước* tệp, gần giống với `-t` và `-X` tương ứng là sắp xếp theo thời gian (time) và phần mở rộng (eXtension). Các tệp có kích thước lớn nhất sẽ xuất hiện trước và kích thước nhỏ nhất sẽ xuất hiện sau cùng. Hãy lưu ý rằng nội dung của các thư mục con sẽ *không* được bao gồm trong thứ tự sắp xếp.

ls -R

Tùy chọn `-R` sẽ sửa đổi lệnh `ls` để hiển thị danh sách *đệ quy*. Điều này có nghĩa là gì?

Đệ quy trong Bash

Đệ quy được định nghĩa là một tình huống khi “một thứ gì đó được định nghĩa dựa trên chính nó”. Đệ quy là một khái niệm rất quan trọng trong khoa học máy tính, nhưng ở đây ý nghĩa của nó sẽ đơn giản hơn nhiều. Hãy xem xét ví dụ từ trước của chúng ta:

```
$ ls ~
```

Documents

Trước đây chúng ta đã biết rằng `user` có một thư mục chính và trong thư mục này có một thư mục con. `ls` cho đến nay chỉ hiển thị cho ta các tệp và thư mục con của một vị trí chứ không thể cho ta biết nội dung của các thư mục con này. Trong các bài học, chúng ta đã sử dụng lệnh `tree` khi muốn hiển thị nội dung của nhiều thư mục. Thật không may, `tree` không phải là một trong những tiện ích cốt lõi của Linux và do đó không phải lúc nào cũng có sẵn. Hãy so sánh đầu ra của `tree` với đầu ra của `ls -R` trong các ví dụ sau:

```
$ tree /home/user
user
├── Documents
│   ├── Mission-Statement
│   └── Reports
│       └── report2018.txt

$ ls -R ~
/home/user/:
Documents

/home/user/Documents:
Mission-Statement
Reports

/home/user/Documents/Reports:
report2018.txt
```

Như có thể thấy, với tùy chọn đệ quy, chúng ta sẽ nhận được một danh sách tệp dài hơn nhiều. Trên thực tế, nó giống như thể chúng ta đã chạy lệnh `ls` trong thư mục chính của `user` và bắt gặp một thư mục con. Sau đó, chúng ta đã vào thư mục con đó và chạy lại lệnh `ls`. Ta đã gặp tệp `Mission-Statement` và một thư mục con khác có tên là `Reports`. Và một lần nữa, ta lại truy cập vào thư mục con và chạy lại lệnh `ls`. Về cơ bản, chạy `ls -R` giống như nói với Bash: “Chạy `ls` tại đây và lặp lại lệnh trong mọi thư mục con mà bạn tìm thấy.”

Đệ quy đặc biệt quan trọng trong các lệnh sửa đổi tệp như sao chép hoặc xóa thư mục. Ví dụ: nếu bạn muốn sao chép thư mục con `Documents`, bạn cần chỉ định một bản sao đệ quy để mở rộng lệnh này cho tất cả các thư mục con.

Bài tập Hướng dẫn

1. Hãy sử dụng cấu trúc tập sau để trả lời ba câu hỏi sau:

```

/
├── etc/
│   ├── network/
│   │   └── interfaces/
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d
│       └── udev.conf
└── home/
    ├── lost+found/
    ├── user/
    │   └── Documents/
    └── michael/
        └── Music/
  
```

◦ Lệnh nào sẽ điều hướng vào thư mục `network` bất kể vị trí hiện tại của bạn là gì?

◦ `user` có thể nhập lệnh nào để điều hướng vào thư mục `Documents` của nó từ `/etc/udev`? Hãy sử dụng đường dẫn ngắn nhất có thể.

◦ `user` có thể nhập lệnh nào để điều hướng vào thư mục `Music` của `michael`? Hãy sử dụng đường dẫn ngắn nhất có thể.

2. Hãy xem xét đầu ra sau của `ls -lh` để trả lời hai câu hỏi tiếp theo. Hãy lưu ý rằng các thư mục sẽ được biểu thị bằng `d` ở đầu dòng.

```

drwxrwxrwx 5 eric eric 4.0K Apr 26 2011 China/
-rwxrwxrwx 1 eric eric 1.5M Jul 18 2011 img_0066.jpg
-rwxrwxrwx 1 eric eric 1.5M Jul 18 2011 img_0067.jpg
  
```

```
-rwxrwxrwx 1 eric eric 1.6M Jul 18 2011 img_0074.jpg
-rwxrwxrwx 1 eric eric 1.8M Jul 18 2011 img_0075.jpg
-rwxrwxrwx 1 eric eric 46K Jul 18 2011 scary.jpg
-rwxrwxrwx 1 eric eric 469K Jan 29 2018 Screenshot from 2017-08-13 21-22-24.png
-rwxrwxrwx 1 eric eric 498K Jan 29 2018 Screenshot from 2017-08-14 21-18-07.png
-rwxrwxrwx 1 eric eric 211K Jan 29 2018 Screenshot from 2018-01-06 23-29-30.png
-rwxrwxrwx 1 eric eric 150K Jul 18 2011 tobermory.jpg
drwxrwxrwx 6 eric eric 4.0K Apr 26 2011 Tokyo/
-rwxrwxrwx 1 eric eric 1.4M Jul 18 2011 Toronto 081.jpg
-rwxrwxrwx 1 eric eric 1.4M Jul 18 2011 Toronto 085.jpg
-rwxrwxrwx 1 eric eric 944K Jul 18 2011 Toronto 152.jpg
-rwxrwxrwx 1 eric eric 728K Jul 18 2011 Toronto 173.jpg
drwxrwxrwx 2 eric eric 4.0K Jun 5 2016 Wallpapers/
```

- Khi chạy lệnh `ls -lrS` thì file nào đứng đầu?

- Hãy mô tả những gì bạn muốn thấy trong đầu ra của `ls -ad */`.

Bài tập Mở rộng

1. Hãy chạy lệnh `ls -lh` trong thư mục chứa các thư mục con. Hãy lưu ý kích thước được liệt kê của các thư mục này. Kích thước của các tệp này trông có hợp lý hay không? Chúng có thể hiện chính xác nội dung của tất cả các tệp bên trong thư mục đó không?

2. Đây là một lệnh mới để thử: `du -h`. Hãy chạy lệnh này và mô tả đầu ra của nó.

3. Trên nhiều hệ thống Linux, bạn có thể nhập `ll` và nhận được kết quả giống như khi bạn nhập `ls -l`. Tuy nhiên, hãy lưu ý rằng `ll` *không* phải là một lệnh. Ví dụ: `man ll` sẽ cho ra thông báo rằng không có mục nhập thủ công nào cho nó. Đây là một ví dụ về *bí danh*. Tại sao bí danh có thể trở nên hữu ích cho người dùng?

Tóm tắt

Trong bài học này, bạn đã học được: * mỗi người dùng Linux sẽ có một thư mục chính,

- có thể truy cập thư mục chính của người dùng hiện tại bằng cách sử dụng `~`,
- bất kỳ đường dẫn tệp nào sử dụng `~` đều được gọi là đường dẫn *tương đối đến trang chủ*.

Bạn cũng đã học về một số cách phổ biến nhất để sửa đổi lệnh `ls`.

-a (all)

in tất cả các tệp/thư mục, bao gồm cả các thành phần ẩn

-d (directories)

liệt kê các thư mục chứ không phải nội dung của chúng

-h (human readable)

in kích thước tệp ở định dạng con người có thể đọc được

-l (long list)

cung cấp thêm chi tiết, mỗi một tệp/thư mục trên mỗi dòng

-r (reverse)

đảo ngược thứ tự sắp xếp

-R (recursive)

liệt kê mọi tệp, bao gồm cả các tệp trong mỗi thư mục con

-S (size)

sắp xếp theo kích thước tệp

-t (time)

sắp xếp theo thời gian sửa đổi

-X (eXtension)

sắp xếp theo phần mở rộng tệp

Đáp án Bài tập Hướng dẫn

1. Hãy sử dụng cấu trúc tập sau để trả lời ba câu hỏi sau:

```
/
├── etc/
│   ├── network/
│   │   └── interfaces/
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d
│       └── udev.conf
└── home/
    ├── lost+found/
    ├── user/
    │   └── Documents/
    └── michael/
        └── Music/
```

- Lệnh nào sẽ điều hướng vào thư mục `network` bất kể vị trí hiện tại của bạn là gì?

```
cd /etc/network
```

- `user` có thể nhập lệnh nào để điều hướng vào thư mục `Documents` của nó từ `/etc/udev`? Hãy sử dụng đường dẫn ngắn nhất có thể.

```
cd ~/Documents
```

- `user` có thể nhập lệnh nào để điều hướng vào thư mục `Music` của `michael`? Hãy sử dụng đường dẫn ngắn nhất có thể.

```
cd ~michael/Music
```

2. Hãy xem xét đầu ra sau của `ls -lh` để trả lời hai câu hỏi tiếp theo. Hãy lưu ý rằng các thư mục

sẽ được biểu thị bằng d ở đầu dòng.

```
drwxrwxrwx 5 eric eric 4.0K Apr 26 2011 China/
-rwxrwxrwx 1 eric eric 1.5M Jul 18 2011 img_0066.jpg
-rwxrwxrwx 1 eric eric 1.5M Jul 18 2011 img_0067.jpg
-rwxrwxrwx 1 eric eric 1.6M Jul 18 2011 img_0074.jpg
-rwxrwxrwx 1 eric eric 1.8M Jul 18 2011 img_0075.jpg
-rwxrwxrwx 1 eric eric 46K Jul 18 2011 scary.jpg
-rwxrwxrwx 1 eric eric 469K Jan 29 2018 Screenshot from 2017-08-13 21-22-24.png
-rwxrwxrwx 1 eric eric 498K Jan 29 2018 Screenshot from 2017-08-14 21-18-07.png
-rwxrwxrwx 1 eric eric 211K Jan 29 2018 Screenshot from 2018-01-06 23-29-30.png
-rwxrwxrwx 1 eric eric 150K Jul 18 2011 tobermory.jpg
drwxrwxrwx 6 eric eric 4.0K Apr 26 2011 Tokyo/
-rwxrwxrwx 1 eric eric 1.4M Jul 18 2011 Toronto 081.jpg
-rwxrwxrwx 1 eric eric 1.4M Jul 18 2011 Toronto 085.jpg
-rwxrwxrwx 1 eric eric 944K Jul 18 2011 Toronto 152.jpg
-rwxrwxrwx 1 eric eric 728K Jul 18 2011 Toronto 173.jpg
drwxrwxrwx 2 eric eric 4.0K Jun 5 2016 Wallpapers/
```

- Khi chạy lệnh `ls -lrS` thì file nào đứng đầu?

Cả ba thư mục đều là 4.0K, đây là kích thước tệp nhỏ nhất. `ls` sau đó sẽ sắp xếp các thư mục theo thứ tự bảng chữ cái theo mặc định. Câu trả lời đúng ở đây là tệp `scary.jpg`.

- Hãy mô tả những gì bạn muốn thấy trong đầu ra của `ls -ad */`.

Lệnh này sẽ hiển thị tất cả các thư mục con, kể cả các thư mục con ẩn.

Đáp án Bài tập Mở rộng

1. Hãy chạy lệnh `ls -lh` trong thư mục chứa các thư mục con. Hãy lưu ý kích thước được liệt kê của các thư mục này. Kích thước của các tập này trông có hợp lý hay không? Chúng có thể hiện chính xác nội dung của tất cả các tập bên trong thư mục đó không?

Không. Mỗi thư mục đều có kích thước tập được liệt kê là 4096 byte. Điều này là do các thư mục ở đây là dạng trừu tượng: chúng không tồn tại dưới dạng cấu trúc cây trên đĩa. Khi bạn thấy một thư mục được liệt kê tức là bạn đang thấy một *liên kết* tới một danh sách các tập. Kích thước của các liên kết này là 4096 byte.

2. Đây là một lệnh mới để thử: `du -h`. Hãy chạy lệnh này và mô tả đầu ra của nó.

Lệnh `du` sẽ hiển thị danh sách tất cả các tập và thư mục, đồng thời cho biết kích thước của từng tập. Ví dụ: `du -s` sẽ hiển thị kích thước tập của tất cả các tập, thư mục và thư mục con cho một vị trí nhất định.

3. Trên nhiều hệ thống Linux, bạn có thể nhập `ll` và nhận được kết quả giống như khi bạn nhập `ls -l`. Tuy nhiên, hãy lưu ý rằng `ll` *không* phải là một lệnh. Ví dụ: `man ll` sẽ cho ra thông báo rằng không có mục nhập thủ công nào cho nó. Đây là một ví dụ về *bí danh*. Tại sao bí danh có thể trở nên hữu ích cho người dùng?

`ll` là *bí danh* của `ls -l`. Trong Bash, chúng ta có thể sử dụng bí danh để đơn giản hóa các lệnh thường được sử dụng. `ll` thường được định sẵn cho bạn trong Linux, nhưng bạn cũng có thể tạo các bí danh của riêng mình.



**Linux
Professional
Institute**

2.4 Tạo, di chuyển và xóa Tập

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 2.4](#)

Khối lượng

2

Các lĩnh vực kiến thức chính

- Tập và thư mục
- Phân biệt chữ hoa chữ thường
- Khớp mẫu vòng

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

- `mv`, `cp`, `rm`, `touch`
- `mkdir`, `rmdir`



2.4 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	2 Tìm đường trong Hệ thống Linux
Mục tiêu:	2.4 Tạo, di chuyển và xóa Tập
Bài:	1 trên 1

Giới thiệu

Trong bài học này, chúng ta sẽ cùng nói về vấn đề quản lý tập và thư mục trên Linux bằng các công cụ dòng lệnh.

Tập là một tập hợp dữ liệu có tên và một tập hợp các thuộc tính. Ví dụ: bạn chuyển một số hình ảnh từ điện thoại sang máy tính và đặt tên mô tả cho chúng, kết quả là bạn sẽ có một loạt các tập hình ảnh trên máy tính của mình. Các tập này có các thuộc tính như thời gian tập được truy cập hoặc sửa đổi lần cuối.

Thư mục là một loại tập đặc biệt được sử dụng để tổ chức các tập. Có thể hình dung các thư mục giống như các bìa hồ sơ được dùng để lưu trữ giấy tờ trong tủ hồ sơ. Khác với các tập hồ sơ giấy, ta có thể dễ dàng đặt các thư mục này bên trong các thư mục khác.

Dòng lệnh là cách hiệu quả nhất để quản lý tập trên hệ thống Linux. Vỏ và các công cụ dòng lệnh có các tính năng giúp người dùng sử dụng dòng lệnh nhanh chóng và dễ dàng hơn so với trình quản lý tập đồ họa.

Trong phần này, chúng ta sẽ sử dụng các lệnh `ls`, `mv`, `cp`, `pwd`, `find`, `touch`, `rm`, `rmdir`, `echo`, `cat`, và `mkdir` để quản lý và sắp xếp các tập và thư mục.

Phân biệt Chữ hoa và Chữ thường

Không giống như Microsoft Windows, tên tệp và thư mục trên hệ thống Linux phải phân biệt chữ hoa và chữ thường. Điều này có nghĩa là tên `/etc/` và `/ETC/` là các thư mục hoàn toàn khác nhau. Hãy thử các lệnh sau:

```
$ cd /
$ ls
bin  dev  home  lib64  mnt  proc  run  srv  tmp  var
boot  etc  lib  media  opt  root  sbin  sys  usr
$ cd ETC
bash: cd: ETC: No such file or directory
$ pwd
/
$ cd etc
$ pwd
/etc
```

`pwd` sẽ cho biết vị trí thư mục hiện tại của bạn. Như có thể thấy, việc thay đổi thành `/ETC` không khả dụng vì không có thư mục như vậy. Nếu thay đổi thành thư mục `/etc` (thư mục có tồn tại) thì lệnh sẽ hoạt động bình thường.

Tạo Thư mục

Lệnh `mkdir` được sử dụng để tạo thư mục.

Hãy tạo một thư mục mới trong thư mục chính:

```
$ cd ~
$ pwd
/home/user
$ ls
Desktop  Documents  Downloads
$ mkdir linux_essentials-2.4
$ ls
Desktop  Documents  Downloads  linux_essentials-2.4
$ cd linux_essentials-2.4
$ pwd
/home/emma/linux_essentials-2.4
```

Trong suốt bài học này, tất cả các lệnh sẽ diễn ra ở trong thư mục này hoặc một trong các thư mục

con của nó.

Để dễ dàng quay lại thư mục của bài học từ bất kỳ vị trí nào khác trong hệ thống tệp của bạn, bạn có thể sử dụng lệnh sau:

```
$ cd ~/linux_essentials-2.4
```

Vỏ sẽ hiểu ký tự `~` là thư mục chính của bạn.

Khi bạn đang ở trong thư mục của bài học, hãy tạo thêm một số thư mục mà chúng ta sẽ sử dụng cho các bài tập. Bạn có thể thêm tất cả các tên thư mục được phân tách bằng dấu cách vào `mkdir`:

```
$ mkdir creating moving copying/files copying/directories deleting/directories
deleting/files globs
mkdir: cannot create directory 'copying/files': No such file or directory
mkdir: cannot create directory 'copying/directories': No such file or directory
mkdir: cannot create directory 'deleting/directories': No such file or directory
mkdir: cannot create directory 'deleting/files': No such file or directory
$ ls
creating globs moving
```

Hãy lưu ý về các thông báo lỗi và việc chỉ có `moving`, `globs` và `creating` được tạo. Các thư mục `copying` và `deleting` chưa tồn tại. Theo mặc định, `mkdir` sẽ không tạo thư mục bên trong một thư mục không tồn tại. Tùy chọn `-p` hoặc `--parents` sẽ hướng dẫn `mkdir` tạo các thư mục mẹ nếu chúng không tồn tại. Hãy thử lệnh `mkdir` tương tự với tùy chọn `-p`:

```
$ mkdir -p creating moving copying/files copying/directories deleting/directories
deleting/files globs
```

Bây giờ thì bạn sẽ không nhận được bất kỳ thông báo lỗi nào nữa. Hãy xem thử xem những thư mục nào hiện đang tồn tại:

```
$ find
.
./creating
./moving
./globs
./copying
./copying/files
./copying/directories
```

```
./deleting
./deleting/directories
./deleting/files
```

Chương trình `find` thường được sử dụng để tìm kiếm các tệp và thư mục; nhưng khi không có bất kỳ tùy chọn nào, chương trình sẽ hiển thị cho bạn danh sách tất cả các tệp, thư mục và thư mục con của thư mục hiện tại.

TIP

Khi liệt kê nội dung của một thư mục bằng `ls`, các tùy chọn `-t` và `-r` sẽ trở nên đặc biệt hữu dụng. Chúng có thể sắp xếp đầu ra theo thời gian (`-t`) và đảo ngược thứ tự sắp xếp (`-r`). Trong trường hợp này, các tệp mới nhất sẽ ở cuối đầu ra.

Tạo Tệp

Thông thường, các tệp sẽ được tạo bởi các chương trình làm việc với dữ liệu được lưu trữ trong tệp. Một tệp trống có thể được tạo bằng lệnh `touch`. Nếu bạn chạy `touch` trên một tệp hiện có, nội dung của tệp sẽ không bị thay đổi nhưng dấu thời gian sửa đổi tệp sẽ được cập nhật.

Hãy chạy lệnh sau để tạo một số tệp cho bài học khớp mẫu khối (globbing):

```
$ touch globs/question1 globs/question2012 globs/question23 globs/question13
globs/question14
$ touch globs/star10 globs/star1100 globs/star2002 globs/star2013
```

Bây giờ, hãy xác minh tất cả các tệp tồn tại trong thư mục `globs`:

```
$ cd globs
$ ls
question1  question14  question23  star1100  star2013
question13 question2012 star10       star2002
```

Hãy lưu ý cách mà `touch` đã tạo các tệp. Bạn có thể xem nội dung của tệp văn bản bằng lệnh `cat`. Hãy thử nó trên một trong những tệp bạn vừa tạo:

```
$ cat question14
```

Vì `touch` tạo ra các tệp trống nên sẽ không có đầu ra. Bạn có thể sử dụng `echo` với `>` để tạo các tệp văn bản đơn giản. Hãy thử:

```
$ echo hello > question15
$ cat question15
hello
```

`echo` sẽ hiển thị văn bản trên dòng lệnh. Ký tự `>` hướng dẫn vỏ ghi đầu ra của lệnh vào tệp đã được chỉ định thay vì vào cửa sổ dòng lệnh. Điều này dẫn đến đầu ra của `echo`, trong trường hợp này là `hello`, sẽ được ghi vào tệp `question15`. Cách sử dụng này không chỉ áp dụng với `echo` mà có thể được sử dụng với bất kỳ lệnh nào.

WARNING

Hãy cẩn thận khi sử dụng `>`! Nếu tệp được đặt tên đã có sẵn từ trước đó thì nó sẽ bị ghi đè!

Đổi tên Tệp

Các tệp có thể được di chuyển và đổi tên bằng lệnh `mv`.

Hãy đặt thư mục làm việc của bạn thành thư mục `moving`:

```
$ cd ~/linux_essentials-2.4/moving
```

Hãy tạo một số tệp để thực hành. Đến bây giờ, bạn đã khá quen thuộc với các lệnh này:

```
$ touch file1 file22
$ echo file3 > file3
$ echo file4 > file4
$ ls
file1 file22 file3 file4
```

Giả sử tên `file22` là một lỗi đánh máy, đáng ra phải là `file2`. Ta sẽ sửa nó bằng lệnh `mv`. Khi đổi tên tệp, đối số đầu tiên sẽ là tên hiện tại, đối số thứ hai là tên mới:

```
$ mv file22 file2
$ ls
file1 file2 file3 file4
```

Hãy cẩn thận với lệnh `mv`. Nếu bạn đổi tên một tệp thành tên của một tệp khác đã có, nó sẽ ghi đè lên tệp đó. Hãy thử kiểm tra điều này với `file3` và `file4`:

```
$ cat file3
```



```
file3
$ cat file4
file4
$ mv file4 file3
$ cat file3
file4
$ ls
file1 file2 file3
```

Hãy chú ý việc nội dung của `file3` bây giờ đã là `file4`. Hãy sử dụng tùy chọn `-i` để `mv` nhắc bạn nếu bạn sắp ghi đè lên một tệp hiện có. Hãy thử như sau:

```
$ touch file4 file5
$ mv -i file4 file3
mv: overwrite 'file3'? y
```

Di chuyển Tệp

Các tệp có thể được di chuyển từ thư mục này sang thư mục khác bằng lệnh `mv`.

Hãy tạo một vài thư mục để di chuyển tệp vào:

```
$ cd ~/linux_essentials-2.4/moving
$ mkdir dir1 dir2
$ ls
dir1 dir2 file1 file2 file3 file5
```

Di chuyển `file1` vào `dir1`:

```
$ mv file1 dir1
$ ls
dir1 dir2 file2 file3 file5
$ ls dir1
file1
```

Hãy lưu ý việc đối số cuối cùng của `mv` chính là thư mục đích. Bất cứ khi nào đối số cuối cùng của `mv` là một thư mục, các tệp sẽ được chuyển vào đó. Ta có thể chỉ định nhiều tệp trong một lệnh `mv`:

```
$ mv file2 file3 dir2
$ ls
```

```
dir1 dir2 file5
$ ls dir2
file2 file3
```

Cũng có thể sử dụng `mv` để di chuyển và đổi tên thư mục. Hãy thử đổi tên `dir1` thành `dir3`:

```
$ ls
dir1 dir2 file5
$ ls dir1
file1
$ mv dir1 dir3
$ ls
dir2 dir3 file5
$ ls dir3
file1
```

Xóa Tập và Thư mục

Lệnh `rm` có thể xóa các tập và thư mục, trong khi lệnh `rmdir` chỉ có thể xóa các thư mục. Hãy dọn sạch thư mục `moving` bằng cách xóa `file5`:

```
$ cd ~/linux_essentials-2.4/moving
$ ls
dir2 dir3 file5
$ rmdir file5
rmdir: failed to remove 'file5': Not a directory
$ rm file5
$ ls
dir2 dir3
```

Theo mặc định, `rmdir` chỉ có thể xóa các thư mục trống, do đó chúng ta sẽ phải sử dụng `rm` để xóa một tập thông thường. Hãy thử xóa thư mục `deleting`:

```
$ cd ~/linux_essentials-2.4/
$ ls
copying creating deleting globs moving
$ rmdir deleting
rmdir: failed to remove 'deleting': Directory not empty
$ ls -l deleting
total 0
```

```
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 directories
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 files
```

Theo mặc định, `rmdir` sẽ từ chối xóa thư mục có dữ liệu. Hãy thử sử dụng `rmdir` để xóa một trong các thư mục con trống của thư mục `deleting`:

```
$ ls -a deleting/files
.  ..
$ rmdir deleting/files
$ ls -l deleting
directories
```

Việc xóa một số lượng lớn tệp hoặc một cấu trúc thư mục sâu với nhiều thư mục con nghe có vẻ hơi mất công nhưng thực ra lại rất dễ dàng. Theo mặc định, `rm` sẽ chỉ hoạt động trên các tệp thông thường. Tùy chọn `-r` được sử dụng để đề lệnh này. Hãy cẩn thận bởi `rm -r` là một con dao hai lưỡi! Khi bạn sử dụng tùy chọn `-r`, `rm` sẽ không chỉ xóa bất kỳ thư mục nào mà còn xóa mọi nội dung bên trong nó, bao gồm cả các thư mục con và nội dung của chúng. Hãy tự kiểm chứng cách `rm -r` hoạt động:

```
$ ls
copying  creating  deleting  globs  moving
$ rm deleting
rm: cannot remove 'deleting': Is a directory
$ ls -l deleting
total 0
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 directories
$ rm -r deleting
$ ls
copying  creating  globs  moving
```

Hãy lưu ý việc `deleting` đã biến mất dù là nó không hề trống? Giống như `mv`, `rm` có tùy chọn `-i` để nhắc bạn trước khi làm bất cứ điều gì. Hãy sử dụng `rm -ri` để xóa các thư mục không còn cần thiết khỏi phần `moving`:

```
$ find
.
./creating
./moving
./moving/dir2
./moving/dir2/file2
```

```

./moving/dir2/file3
./moving/dir3
./moving/dir3/file1
./globs
./globs/question1
./globs/question2012
./globs/question23
./globs/question13
./globs/question14
./globs/star10
./globs/star1100
./globs/star2002
./globs/star2013
./globs/question15
./copying
./copying/files
./copying/directories
$ rm -ri moving
rm: descend into directory 'moving'? y
rm: descend into directory 'moving/dir2'? y
rm: remove regular empty file 'moving/dir2/file2'? y
rm: remove regular empty file 'moving/dir2/file3'? y
rm: remove directory 'moving/dir2'? y
rm: descend into directory 'moving/dir3'? y
rm: remove regular empty file 'moving/dir3/file1'? y
rm: remove directory 'moving/dir3'? y
rm: remove directory 'moving'? y

```

Sao chép Tập và Thư mục

Lệnh `cp` được sử dụng để sao chép tập và thư mục. Hãy thử sao chép một số tập vào thư mục `copy`:

```

$ cd ~/linux_essentials-2.4/copying
$ ls
directories  files
$ cp /etc/nsswitch.conf files/nsswitch.conf
$ cp /etc/issue /etc/hostname files

```

Nếu đối số cuối cùng là một thư mục, `cp` sẽ tạo một bản sao của các đối số trước đó bên trong thư mục. Giống như `mv`, lệnh này có thể chỉ định nhiều tập cùng một lúc, miễn là đích đến là một thư mục.

Khi cả hai toán hạng của `cp` đều là tệp và cả hai tệp đều tồn tại, `cp` sẽ ghi đè lên tệp thứ hai bằng một bản sao của tệp đầu tiên. Hãy thực hành điều này bằng cách ghi đè tệp `hostname` lên tệp `issue`:

```
$ cd ~/linux_essentials-2.4/copying/files
$ ls
hostname  issue  nsswitch.conf
$ cat hostname
mycomputer
$ cat issue
Debian GNU/Linux 9 \n \l

$ cp hostname issue
$ cat issue
mycomputer
```

Bây giờ, hãy thử tạo một bản sao của thư mục `files` trong thư mục `directories`:

```
$ cd ~/linux_essentials-2.4/copying
$ cp files directories
cp: omitting directory 'files'
```

Như có thể thấy, `cp` theo mặc định chỉ hoạt động trên các tệp riêng lẻ. Để sao chép một thư mục, ta phải sử dụng tùy chọn `-r`. Hãy nhớ rằng tùy chọn `-r` sẽ khiến `cp` cũng sao chép cả nội dung của thư mục bạn đang sao chép:

```
$ cp -r files directories
$ find
.
./files
./files/nsswitch.conf
./files/fstab
./files/hostname
./directories
./directories/files
./directories/files/nsswitch.conf
./directories/files/fstab
./directories/files/hostname
```

Bạn có chú ý thấy khi một thư mục hiện có được sử dụng làm thư mục đích, `cp` sẽ tạo một bản sao của thư mục nguồn bên trong thư mục đó không? Nếu đích đến không tồn tại, nó sẽ tạo ra bản sao

và lấp đầy nó với nội dung của thư mục nguồn:

```
$ cp -r files files2
$ find
.
./files
./files/nsswitch.conf
./files/fstab
./files/hostname
./directories
./directories/files
./directories/files/nsswitch.conf
./directories/files/fstab
./directories/files/hostname
./files2
./files2/nsswitch.conf
./files2/fstab
./files2/hostname
```

Khớp mẫu khối

Cái thường được gọi là khớp mẫu khối (globbing) là một ngôn ngữ khớp mẫu đơn giản. Các vở dòng lệnh trên hệ thống Linux sử dụng ngôn ngữ này để chỉ các nhóm tệp có tên khớp với một mẫu cụ thể. POSIX.1-2017 chỉ định các ký tự khớp mẫu sau:

Khớp với bất kỳ số nào của bất kỳ ký tự nào, kể cả khi không có ký tự nào

?

Khớp với bất kỳ một ký tự nào

[]

Khớp với một lớp ký tự

Điều này có nghĩa là bạn có thể yêu cầu vở khớp với một mẫu thay vì một chuỗi văn bản. Thông thường, người dùng Linux có thể chỉ định nhiều tệp bằng một khối khớp mẫu thay vì nhập từng tên tệp. Hãy thử chạy các lệnh sau:

```
$ cd ~/linux_essentials-2.4/globs
$ ls
question1  question14  question2012  star10  star2002
```

```
question13 question15 question23 star1100 star2013
$ ls star1*
star10 star1100
$ ls star*
star10 star1100 star2002 star2013
$ ls star2*
star2002 star2013
$ ls star2*2
star2002
$ ls star2013*
star2013
```

Vỏ sẽ mở rộng `*` thành một số lượng bất kỳ của bất cứ một cái gì; do đó, vỏ của bạn sẽ diễn giải `star*` có nghĩa là bất kỳ thứ gì trong ngữ cảnh có liên quan bắt đầu bằng `star`. Khi bạn chạy lệnh `ls star*`, vỏ sẽ không chạy chương trình `ls` với đối số là `star*`; nó sẽ tìm các tệp trong thư mục hiện tại khớp với mẫu `star*` (bao gồm chỉ `star`) và biến từng tệp khớp với mẫu thành một đối số cho `ls`:

```
$ ls star*
```

theo như `ls` tương đương với

```
$ ls star10 star1100 star2002 star2013
```

Ký tự `*` không có ý nghĩa gì đối với `ls`. Để chứng minh cho điều này, hãy chạy lệnh sau:

```
$ ls star\*
ls: cannot access star*: No such file or directory
```

Khi bạn đặt `\` trước một ký tự, bạn đang nói với vỏ rằng đừng diễn giải ký tự đó. Khi đặt như vậy, bạn đang muốn `ls` có đối số là `star*` thay vì kết quả của khối khớp mẫu `star*`.

? có thể mở rộng thành bất kỳ ký tự đơn nào. Hãy thử các lệnh sau để kiểm chứng:

```
$ ls
question1 question14 question2012 star10 star2002
question13 question15 question23 star1100 star2013
$ ls question?
question1
$ ls question1?
```

```
question13 question14 question15
$ ls question?3
question13 question23
$ ls question13?
ls: cannot access question13?: No such file or directory
```

Dấu ngoặc `[]` được sử dụng để khớp với các phạm vi hoặc lớp ký tự. Các dấu ngoặc `[]` hoạt động giống như trong các biểu thức chính quy POSIX, ngoại trừ việc với các khối khớp mẫu thì `^` được sử dụng thay cho `!`.

Hãy tạo một số tệp để thử nghiệm:

```
$ mkdir brackets
$ cd brackets
$ touch file1 file2 file3 file4 filea fileb filec file5 file6 file7
```

Các phạm vi trong dấu ngoặc `[]` sẽ được thể hiện bằng dấu `-`:

```
$ ls
file1 file2 file3 file4 file5 file6 file7 filea fileb filec
$ ls file[1-2]
file1 file2
$ ls file[1-3]
file1 file2 file3
```

Nhiều phạm vi có thể được chỉ định:

```
$ ls file[1-25-7]
file1 file2 file5 file6 file7
$ ls file[1-35-6a-c]
file1 file2 file3 file5 file6 filea fileb filec
```

Dấu ngoặc vuông cũng có thể được sử dụng để khớp với một bộ ký tự cụ thể.

```
$ ls file[1a5]
file1 file5 filea
```

Bạn cũng có thể sử dụng ký tự `^` làm ký tự đầu tiên để khớp với mọi thứ trừ một số ký tự nhất định.


```
$ ls file[^a]
file1 file2 file3 file4 file5 file6 file7 fileb filec
```

Điều cuối cùng chúng ta sẽ đề cập đến trong bài học này là các lớp ký tự. Để khớp với một lớp ký tự, ta sẽ sử dụng `[:classname:]`. Ví dụ: để sử dụng lớp ký tự số khớp với các chữ số, bạn sẽ làm như sau:

```
$ ls file[[:digit:]]
file1 file2 file3 file4 file5 file6 file7
$ touch file1a file11
$ ls file[[:digit:]]a
file1 file2 file3 file4 file5 file6 file7 filea
$ ls file[[:digit:]]a
file1a
```

Khối khớp mẫu `file[[:digit:]]a` khớp với `file` theo sau bởi một ký tự số (digit) hoặc `a`. Các lớp ký tự được hỗ trợ sẽ tùy thuộc vào ngôn ngữ hiện tại của bạn. POSIX yêu cầu các lớp ký tự sau cho tất cả các ngôn ngữ:

[:alnum:]

Chữ và số.

[:alpha:]

Chữ hoa hoặc chữ thường.

[:blank:]

Các khoảng trắng và tabs.

[:cntrl:]

Ký tự điều khiển; ví dụ: xóa lùi, chuông, NAK, thoát.

[:chữ số:]

Chữ số (0123456789).

[:graph:]

Ký tự đồ họa (tất cả các ký tự ngoại trừ `ctrl` và ký tự khoảng trắng)

[:lower:]

Chữ thường (a - z).

[:print:]

Các ký tự có thể in được (alnum, punct và ký tự khoảng trắng).

[:punct:]

Các ký tự dấu câu, tức !, &, ".

[:dấu cách:]

Các ký tự khoảng trắng; ví dụ: tab, dấu cách, dòng mới.

[:upper:]

Chữ hoa (A-Z).

[:xdigit:]

Số thập lục phân (thường là 0123456789abcdefABCDEF).

Bài tập Hướng dẫn

1. Cho các thông tin sau, hãy chọn các thư mục sẽ được tạo bằng lệnh `mkdir -p /tmp/outfiles/text/today /tmp/infiles/text/today`

```
$ pwd
/tmp
$ find
.
./outfiles
./outfiles/text
```

/tmp	
/tmp/outfiles	
/tmp/outfiles/text	
/tmp/outfiles/text/today	
/tmp/infiles	
/tmp/infiles/text	
/tmp/infiles/text/today	

2. Tùy chọn `-v` trong các lệnh `mkdir`, `rm` và `cp` sẽ làm gì?

3. Điều gì sẽ xảy ra nếu bạn vô tình sao chép ba tệp trên cùng một dòng lệnh vào một tệp đã tồn tại thay vì một thư mục?

4. Điều gì sẽ xảy ra khi bạn sử dụng `mv` để di chuyển một thư mục vào chính nó?

5. Bạn sẽ xóa tất cả các tệp trong thư mục hiện tại bắt đầu bằng `old` như thế nào?

6. Tệp `log_[a-z]_201?_*_01.txt` khớp với tệp nào sau đây?

<code>log_3_2017_Jan_01.txt</code>	
<code>log+_2017_Feb_01.txt</code>	

log_b_2007_Mar_01.txt	
log_f_201A_Wednesday_01.txt	

7. Hãy tạo một vài khối khớp mẫu để khớp với danh sách tên tệp sau:

```
doc100
```

```
doc200
```

```
doc301
```

```
doc401
```

Bài tập Mở rộng

1. Hãy sử dụng trang hướng dẫn của `cp` để tìm hiểu cách tạo một bản sao của tệp có các quyền và thời gian sửa đổi khớp với bản gốc.

2. Lệnh `rmdir -p` thực hiện tác vụ gì? Hãy thử và giải thích xem nó khác với `rm -r` như thế nào.

3. KHÔNG ĐƯỢC THỰC SỰ THỰC HIỆN LỆNH NÀY: Bạn nghĩ `rm -ri /*` sẽ thực hiện tác vụ gì? (ĐỪNG CỐ GẮNG THỬ!)

4. Ngoài việc sử dụng `-i`, ta có thể ngăn `mv` ghi đè lên các tệp đích không?

5. Hãy giải thích lệnh `cp -u`.

Tóm tắt

Môi trường dòng lệnh Linux cung cấp các công cụ để quản lý tệp. Một số lệnh thường được sử dụng là `cp`, `mv`, `mkdir`, `rm` và `rmdir`. Những công cụ này khi kết hợp với các khối khớp mã sẽ cho phép người dùng hoàn thành rất nhiều công việc một cách nhanh chóng.

Nhiều lệnh sẽ có tùy chọn `-i` để cảnh báo bạn trước khi thực hiện bất kỳ điều gì. Lời nhắc nhở này có thể giúp bạn tránh được rất nhiều rắc rối nếu bạn nhập sai lệnh. Rất nhiều lệnh có tùy chọn `-r`. Tùy chọn `-r` thường thường được hiểu theo nghĩa là đệ quy. Trong toán học và khoa học máy tính, một hàm đệ quy là một hàm sử dụng chính nó trong định nghĩa của nó. Khi nói đến các công cụ dòng lệnh, điều này thường có nghĩa là áp dụng lệnh cho một thư mục và toàn bộ nội dung ở bên trong đó.

Các lệnh đã được dùng trong bài học này:

cat

Đọc và xuất nội dung của một tệp.

cp

Sao chép tệp hoặc thư mục.

echo

Xuất một chuỗi.

find

Duyệt qua cây hệ thống tệp và tìm kiếm các tệp khớp với một bộ tiêu chí cụ thể.

ls

Hiển thị thuộc tính của tệp và thư mục cũng như liệt kê nội dung của thư mục.

mkdir

Tạo một thư mục mới

mv

Di chuyển hoặc đổi tên tệp hoặc thư mục.

pwd

Xuất thư mục làm việc hiện tại.

rm

Xóa tệp hoặc thư mục.

rmdir

Xóa thư mục.

touch

Tạo một tệp trống hoặc cập nhật ngày sửa đổi của tệp hiện có.

Đáp án Bài tập Hướng dẫn

1. Cho các thông tin sau, hãy chọn các thư mục sẽ được tạo bằng lệnh `mkdir -p /tmp/outfiles/text/today /tmp/infiles/text/today`

```
$ pwd
/tmp
$ find
.
./outfiles
./outfiles/text
```

Các thư mục được đánh dấu sẽ được tạo. Các thư mục `/tmp`, `/tmp/outfiles` và `/tmp/outfiles/text` đã tồn tại, vì vậy `mkdir` sẽ bỏ qua chúng.

<code>/tmp</code>	
<code>/tmp/outfiles</code>	
<code>/tmp/outfiles/text</code>	
<code>/tmp/outfiles/text/today</code>	X
<code>/tmp/infiles</code>	X
<code>/tmp/infiles/text</code>	X
<code>/tmp/infiles/text/today</code>	X

2. Tùy chọn `-v` trong các lệnh `mkdir`, `rm` và `cp` sẽ làm gì?

Thông thường, `-v` sẽ bật chế độ hiển thị đầu ra chi tiết. Nó khiến các chương trình tương ứng xuất ra thông tin chi tiết về những gì chúng đang thực hiện trong thời gian thực hiện chính những thứ đó:`

```
$ rm -v a b
removed 'a'
removed 'b'
$ mv -v a b
'a' -> 'b'
$ cp -v b c
'b' -> 'c'
```

3. Điều gì sẽ xảy ra nếu bạn vô tình sao chép ba tệp trên cùng một dòng lệnh vào một tệp đã tồn

tại thay vì một thư mục?

cp sẽ từ chối thực hiện bất cứ một tác vụ nào và đưa ra thông báo lỗi:

```
$ touch a b c d
$ cp a b c d
cp: target 'd' is not a directory
```

4. Điều gì sẽ xảy ra khi bạn sử dụng mv để di chuyển một thư mục vào chính nó?

Bạn sẽ nhận được thông báo lỗi cho biết mv không thể làm được điều đó.

```
$ mv a a
mv: cannot move 'a' to a subdirectory of itself, 'a/a'
```

5. Bạn sẽ xóa tất cả các tệp trong thư mục hiện tại bắt đầu bằng old như thế nào?

Sử dụng khối khớp mẫu old* với rm:

```
$ rm old*
```

6. Tệp log_[a-z]_201?*_01.txt khớp với tệp nào sau đây?

log_3_2017_Jan_01.txt	
log+_2017_Feb_01.txt	
log_b_2007_Mar_01.txt	
log_f_201A_Wednesday_01.txt	X

```
$ ls log_[a-z]_201?*_01.txt
log_f_201A_Wednesday_01.txt
```

log_[a-z] khớp với log_ theo sau bởi bất kỳ chữ cái viết thường nào, vì vậy cả chuỗi log_f_201A_Wednesday_01.txt và log_b_2007_Mar_01.txt đều khớp. _201? khớp với bất kỳ ký tự đơn nào, vì vậy chỉ có log_f_201A_Wednesday_01.txt khớp. Cuối cùng, *_01.txt khớp với mọi chuỗi kết thúc bằng _01.txt, vì vậy tùy chọn còn lại của chúng ta khớp.

7. Hãy tạo một vài khối khớp mẫu để khớp với danh sách tên tệp sau:

```
doc100  
doc200  
doc301  
doc401
```

Ở đây ta có nhiều giải pháp. Dưới đây là một vài ví dụ trong số đó:

```
doc*  
doc[1-4]*  
doc??  
doc[1-4]??
```

Đáp án Bài tập Mở rộng

- Hãy sử dụng trang hướng dẫn của `cp` để tìm hiểu cách tạo một bản sao của tệp có các quyền và thời gian sửa đổi khớp với bản gốc.

Bạn sẽ sử dụng tùy chọn `-p`. Từ trang hướng dẫn:

```
$ man cp
-p      same as --preserve=mode,ownership,timestamps
--preserve[=ATTR_LIST]
        preserve the specified attributes (default: mode,ownership,time-
        stamps), if possible additional attributes: context, links,
        xattr, all
```

- Lệnh `rmdir -p` thực hiện tác vụ gì? Hãy thử và giải thích xem nó khác với `rm -r` như thế nào.

Nó khiến `rmdir` hoạt động tương tự như `mkdir -p`. Nếu đi qua một cây thư mục trống, nó sẽ xóa tất cả.

```
$ find
.
./a
./a/b
./a/b/c
$ rmdir -p a/b/c
$ ls
```

- KHÔNG ĐƯỢC THỰC SỰ THỰC HIỆN LỆNH NÀY:** Bạn nghĩ `rm -ri /*` sẽ thực hiện tác vụ gì? (ĐỪNG CỐ GẮNG THỬ!)

Nó sẽ xóa tất cả các tệp và thư mục mà tài khoản người dùng của bạn có thể ghi. Điều này bao gồm bất kỳ hệ thống tệp mạng nào.

- Ngoài việc sử dụng `-i`, có thể ngăn `mv` ghi đè lên các tệp đích không?

Có, tùy chọn `-n` hoặc `--no-clobber` ngăn `mv` ghi đè tệp.

```
$ cat a
a
$ cat b
b
```

```
$ mv -n a b
$ cat b
b
```

5. Hãy giải thích lệnh `cp -u`.

Tùy chọn `-u` khiến `cp` chỉ sao chép tệp nếu đích đến bị thiếu hoặc cũ hơn tệp nguồn.

```
$ ls -l
total 24K
drwxr-xr-x 123 emma student 12K Feb  2 05:34 ..
drwxr-xr-x  2 emma student 4.0K Feb  2 06:56 .
-rw-r--r--  1 emma student  2 Feb  2 06:56 a
-rw-r--r--  1 emma student  2 Feb  2 07:00 b
$ cat a
a
$ cat b
b
$ cp -u a b
$ cat b
b
$ cp -u a c
$ ls -l
total 12
-rw-r--r-- 1 emma student 2 Feb  2 06:56 a
-rw-r--r-- 1 emma student 2 Feb  2 07:00 b
-rw-r--r-- 1 emma student 2 Feb  2 07:00 c
```



Chủ đề 3: Sức mạnh của Dòng lệnh



**Linux
Professional
Institute**

3.1 Lưu trữ Tập trên Dòng lệnh

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 3.1](#)

Khối lượng

2

Các lĩnh vực kiến thức chính

- Tập, thư mục
- Lưu trữ, nén

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

- `tar`
- Các tùy chọn phổ biến của `tar`
- `gzip`, `bzip2`, `xz`
- `zip`, `unzip`



3.1 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	3 Sức mạnh của Dòng lệnh
Mục tiêu:	3.1 Lưu trữ Tập trên Dòng lệnh
Bài:	1 trên 1

Giới thiệu

Nén dữ liệu được sử dụng để giảm lượng không gian mà một tập dữ liệu nhất định chiếm. Thông thường, nén dữ liệu được dùng để giảm lượng không gian cần thiết để lưu trữ một tệp tin. Một tác dụng phổ biến khác của nó là để giảm lượng dữ liệu được gửi qua kết nối mạng.

Nén dữ liệu hoạt động theo nguyên tắc thay thế các mẫu lặp đi lặp lại trong dữ liệu. Giả sử bạn có một cuốn tiểu thuyết. Trong đó, một số từ sẽ được sử dụng rất nhiều và cũng có nhiều ký tự, chẳng hạn như từ “the”. Bạn có thể giảm đáng kể kích thước của cuốn tiểu thuyết nếu bạn thay thế các từ và mẫu nhiều ký tự phổ biến trong cuốn sách này bằng các từ thay thế chỉ có một ký tự. Ví dụ: thay thế từ “the” bằng một chữ cái Hy Lạp không được sử dụng ở những nơi khác trong văn bản. Thuật toán nén dữ liệu cũng hoạt động tương tự như vậy nhưng phức tạp hơn.

Nén dữ liệu có hai loại là nén *không tổn hao* và nén *có tổn hao*. Mọi thứ được nén bằng thuật toán không tổn hao có thể được giải nén trở lại dạng ban đầu. Dữ liệu được nén bằng thuật toán có tổn hao sẽ không thể được khôi phục. Các thuật toán tổn hao thường được sử dụng cho hình ảnh, video và âm thanh mà con người không thể nhận thấy khi chất lượng bị giảm sút, không liên quan đến bối cảnh hoặc khi dung lượng hoặc thông lượng mạng được tiết kiệm quan trọng hơn tổn hao đó.

Các công cụ lưu trữ được sử dụng để gộp các tệp và thư mục vào một tệp duy nhất. Một số cách sử dụng phổ biến của các công cụ này là sao lưu, đóng gói mã nguồn phần mềm và lưu giữ dữ liệu.

Lưu trữ và nén dữ liệu thường đi đôi với nhau. Một số công cụ lưu trữ thậm chí còn mặc định nén nội dung của chúng. Những công cụ khác có thể tùy chọn nén nội dung hoặc không. Một vài công cụ lưu trữ phải được sử dụng cùng với các công cụ nén độc lập nếu bạn muốn nén nội dung.

Công cụ phổ biến nhất để lưu trữ tệp trên hệ thống Linux là `tar`. Hầu hết các bản phân phối Linux đều có phiên bản GNU của `tar`, vì vậy đây là sẽ phiên bản được sử dụng trong bài học này. Bản thân `tar` chỉ quản lý việc lưu trữ tệp chứ không nén chúng.

Có rất nhiều công cụ nén có sẵn trên Linux. Một số định dạng không tốn hao phổ biến là `bzip2`, `gzip` và `xz`. Bạn sẽ tìm thấy cả ba trên hầu hết các hệ thống. Bạn có thể bắt gặp một hệ thống cũ hoặc rất giản tiện mà trong đó `xz` hoặc `bzip` không được cài đặt. Nếu trở thành người dùng Linux phổ thông, bạn có thể sẽ gặp các tệp được nén bằng cả ba công cụ trên. Cả ba đều sử dụng các thuật toán khác nhau; do đó, một tệp được nén bằng công cụ này sẽ không thể được giải nén bằng công cụ kia. Công cụ nén cũng có sự đánh đổi. Nếu bạn muốn tỉ lệ nén cao thì quá trình nén và giải nén file sẽ lâu hơn. Điều này là do độ nén cao hơn đòi hỏi nhiều công việc hơn để tìm các mẫu phức tạp hơn. Tất cả các công cụ này đều có thể nén dữ liệu nhưng không thể tạo các kho lưu trữ chứa nhiều tệp.

Các công cụ nén độc lập thường không khả dụng trên các hệ thống Windows. Các công cụ lưu trữ và nén của Windows thường đi kèm với nhau. Hãy ghi nhớ điều này nếu bạn có hệ thống Linux và Windows cần chia sẻ tệp cho nhau.

Các hệ thống Linux cũng có các công cụ để xử lý các tệp `.zip` thường được sử dụng trên hệ thống Windows. Chúng được gọi là `zip` và `unzip`. Các công cụ này không được cài đặt mặc định trên tất cả các hệ thống; vì vậy, nếu cần sử dụng chúng, bạn có thể sẽ phải cài đặt chúng. Điều may mắn ở đây là chúng thường được tìm thấy trong kho gói của các bản phân phối.

Công cụ Nén

Dung lượng ổ đĩa được tiết kiệm bằng cách nén tệp dựa trên một số yếu tố như bản chất của dữ liệu bạn đang nén, thuật toán được sử dụng để nén dữ liệu và mức độ nén. Không phải thuật toán nào cũng sẽ hỗ trợ nhiều mức nén khác nhau.

Hãy cùng bắt đầu với việc thiết lập một số tệp thử nghiệm để nén:

```
$ mkdir ~/linux_essentials-3.1
$ cd ~/linux_essentials-3.1
$ mkdir compression archiving
```



```
$ cd compression
$ cat /etc/* > bigfile 2> /dev/null
```

Bây giờ, chúng ta sẽ tạo ba bản sao của tệp này:

```
$ cp bigfile bigfile2
$ cp bigfile bigfile3
$ cp bigfile bigfile4
$ ls -lh
total 2.8M
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile2
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile3
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile4
```

Bây giờ, chúng ta sẽ nén các tệp bằng các công cụ nén đã được nhắc đến ở trên:

```
$ bzip2 bigfile2
$ gzip bigfile3
$ xz bigfile4
$ ls -lh
total 1.2M
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile
-rw-r--r-- 1 emma emma 170K Jun 23 08:08 bigfile2.bz2
-rw-r--r-- 1 emma emma 179K Jun 23 08:08 bigfile3.gz
-rw-r--r-- 1 emma emma 144K Jun 23 08:08 bigfile4.xz
```

Hãy so sánh kích thước của tệp nén với tệp không bị nén có tên `bigfile`. Cũng hãy lưu ý cách các công cụ nén đã thêm phần mở rộng vào tên tệp và xóa các tệp không nén.

Hãy sử dụng `bunzip2`, `gunzip` hoặc `unxz` để giải nén các tệp:

```
$ bunzip2 bigfile2.bz2
$ gunzip bigfile3.gz
$ unxz bigfile4.xz
$ ls -lh
total 2.8M
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile2
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile3
-rw-r--r-- 1 emma emma 712K Jun 23 08:20 bigfile4
```

Một lần nữa, hãy lưu ý rằng bây giờ tệp nén sẽ bị xóa sau khi được giải nén.

Một số công cụ nén có hỗ trợ nhiều mức nén khác nhau. Mức nén cao hơn thường yêu cầu nhiều bộ nhớ và chu kỳ CPU hơn, nhưng kết quả là tệp nén sẽ nhỏ hơn. Đối với các cấp thấp hơn thì ngược lại. Dưới đây là minh họa với `xz` và `gzip`:

```
$ cp bigfile bigfile-gz1
$ cp bigfile bigfile-gz9
$ gzip -1 bigfile-gz1
$ gzip -9 bigfile-gz9
$ cp bigfile bigfile-xz1
$ cp bigfile bigfile-xz9
$ xz -1 bigfile bigfile-xz1
$ xz -9 bigfile bigfile-xz9
$ ls -lh bigfile bigfile-* *
total 3.5M
-rw-r--r-- 1 emma emma 712K Jun 23 08:08 bigfile
-rw-r--r-- 1 emma emma 205K Jun 23 13:14 bigfile-gz1.gz
-rw-r--r-- 1 emma emma 178K Jun 23 13:14 bigfile-gz9.gz
-rw-r--r-- 1 emma emma 156K Jun 23 08:08 bigfile-xz1.xz
-rw-r--r-- 1 emma emma 143K Jun 23 08:08 bigfile-xz9.xz
```

Không cần thiết phải giải nén một tệp mỗi khi bạn sử dụng nó. Các công cụ nén thường đi kèm với các phiên bản đặc biệt của các công cụ phổ biến được sử dụng để đọc tệp văn bản. Ví dụ: `gzip` có một phiên bản của `cat`, `grep`, `diff`, `less`, `more` và một vài phiên bản khác. Đối với `gzip`, các công cụ sẽ có tiền tố là `z`, với `bzip2` là `bz` và `xz` là `xz`. Dưới đây là một ví dụ về việc sử dụng `zcat` để đọc hiển thị tệp được nén bằng `gzip`:

```
$ cp /etc/hosts ./
$ gzip hosts
$ zcat hosts.gz
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Công cụ Lưu trữ

Chương trình `tar` có lẽ là công cụ lưu trữ được sử dụng rộng rãi nhất trên các hệ thống Linux.

Trong trường hợp bạn đang thắc mắc tại sao nó được đặt tên như vậy, thì nó là viết tắt của “tape archive” (lưu trữ dạng băng). Các tệp được tạo bằng tar thường được gọi là *bóng tar*. Việc các ứng dụng được phân phối dưới dạng mã nguồn ở dạng quả bóng tar là rất phổ biến. Phiên bản GNU của tar mà các bản phân phối Linux cung cấp có rất nhiều tùy chọn. Bài học này sẽ đề cập đến các tùy chọn được sử dụng phổ biến nhất.

Hãy bắt đầu bằng cách tạo một kho lưu trữ các tệp để nén:

```
$ cd ~/linux_essentials-3.1
$ tar cf archiving/3.1.tar compression
```

Tùy chọn `c` sẽ hướng dẫn tar tạo tệp lưu trữ mới và tùy chọn `f` là tên của tệp cần tạo. Đối số ngay sau các tùy chọn sẽ luôn là tên của tệp phải nén. Phần còn lại của các đối số là đường dẫn đến bất kỳ tệp hoặc thư mục nào bạn muốn thêm vào, liệt kê hoặc trích xuất từ tệp. Trong ví dụ này, chúng ta đang thêm thư mục `compression` và tất cả nội dung của nó vào kho lưu trữ.

Để xem nội dung của bóng tar, hãy sử dụng tùy chọn `t` của tar:

```
$ tar -tf 3.1.tar
compression/
compression/bigfile-xz1.xz
compression/bigfile-gz9.gz
compression/hosts.gz
compression/bigfile2
compression/bigfile
compression/bigfile-gz1.gz
compression/bigfile-xz9.xz
compression/bigfile3
compression/bigfile4
```

Hãy lưu ý việc các tùy chọn được bắt đầu bằng `-`. Không giống như hầu hết các chương trình, với tar, `-` không phải là bắt buộc khi chỉ định các tùy chọn, dù nếu được sử dụng thì nó cũng sẽ không gây ảnh hưởng gì.

NOTE

Bạn có thể sử dụng tùy chọn `-v` để cho phép tar xuất tên của các tệp mà nó nén khi tạo hoặc giải nén một kho lưu trữ.

Bây giờ, hãy giải nén tệp:

```
$ cd ~/linux_essentials-3.1/archiving
$ ls
```

```
3.1.tar
$ tar xf 3.1.tar
$ ls
3.1.tar  compression
```

Giả sử bạn chỉ cần một tệp trong kho lưu trữ. Nếu đúng như vậy, bạn có thể chỉ định tệp đó sau tên tệp của kho lưu trữ. Bạn có thể chỉ định nhiều tệp nếu cần:

```
$ cd ~/linux_essentials-3.1/archiving
$ rm -rf compression
$ ls
3.1.tar
$ tar xvf 3.1.tar compression/hosts.gz
compression/
compression/bigfile-xz1.xz
compression/bigfile-gz9.gz
compression/hosts.gz
compression/bigfile2
compression/bigfile
compression/bigfile-gz1.gz
compression/bigfile-xz9.xz
compression/bigfile3
compression/bigfile4
$ ls
3.1.tar  compression
$ ls compression
hosts.gz
```

Ngoại trừ các đường dẫn tuyệt đối (đường dẫn bắt đầu bằng /), các tệp tar sẽ bảo tồn toàn bộ đường dẫn tới các tệp khi chúng được tạo. Vì tệp 3.1.tar được tạo với một thư mục duy nhất nên thư mục đó sẽ được tạo tương ứng với thư mục làm việc hiện tại của bạn khi được giải nén. Một ví dụ khác sẽ làm rõ việc này:

```
$ cd ~/linux_essentials-3.1/archiving
$ rm -rf compression
$ cd ../compression
$ tar cf ../tar/3.1-nodir.tar *
$ cd ../archiving
$ mkdir untar
$ cd untar
$ tar -xf ../3.1-nodir.tar
$ ls
```

```
bigfile  bigfile3  bigfile-gz1.gz  bigfile-xz1.xz  hosts.gz
bigfile2 bigfile4  bigfile-gz9.gz  bigfile-xz9.xz
```

TIP Nếu muốn sử dụng đường dẫn tuyệt đối trong tệp `tar`, bạn phải sử dụng tùy chọn `P`. Hãy lưu ý rằng tùy chọn này có thể ghi đè lên các tệp quan trọng và có thể gây ra lỗi trên hệ thống của bạn.

Chương trình `tar` cũng có thể quản lý quá trình nén và giải nén tài liệu lưu trữ một cách nhanh chóng. `tar` làm như vậy bằng cách gọi một trong các công cụ nén đã được nhắc đến ở trên. Việc này đơn giản là thêm một tùy chọn phù hợp với thuật toán nén. Những tùy chọn được sử dụng phổ biến nhất lần lượt là `j`, `J` và `z` cho `bzip2`, `xz` và `gzip`. Dưới đây là các ví dụ về việc sử dụng các thuật toán nói trên:

```
$ cd ~/linux_essentials-3.1/compression
$ ls
bigfile  bigfile3  bigfile-gz1.gz  bigfile-xz1.xz  hosts.gz
bigfile2 bigfile4  bigfile-gz9.gz  bigfile-xz9.xz
$ tar -czf gzip.tar.gz bigfile bigfile2 bigfile3
$ tar -cjf bzip2.tar.bz2 bigfile bigfile2 bigfile3
$ tar -cJf xz.tar.xz bigfile bigfile2 bigfile3
$ ls -l | grep tar
-rw-r--r-- 1 emma emma 450202 Jun 27 05:56 bzip2.tar.bz2
-rw-r--r-- 1 emma emma 548656 Jun 27 05:55 gzip.tar.gz
-rw-r--r-- 1 emma emma 147068 Jun 27 05:56 xz.tar.xz
```

Hãy lưu ý rằng trong ví dụ này, các tệp `.tar` có kích thước khác nhau. Điều này cho thấy chúng đã được nén thành công. Nếu bạn tạo các kho lưu trữ `.tar` được nén, bạn phải luôn thêm phần mở rộng tệp thứ hai biểu thị thuật toán bạn đã sử dụng. Chúng lần lượt là `.xz`, `.bz` và `.gz` cho `xz`, `bzip2` và `gzip`. Đôi khi các phần mở rộng rút gọn như `.tgz` sẽ được sử dụng.

Có thể thêm tệp vào kho lưu trữ `tar` chưa nén hiện có. Ta có thể sử dụng tùy chọn `u` để thực hiện việc này. Nếu bạn cố gắng thêm vào một lưu trữ nén, bạn sẽ gặp lỗi.

```
$ cd ~/linux_essentials-3.1/compression
$ ls
bigfile  bigfile3  bigfile-gz1.gz  bigfile-xz1.xz  bzip2.tar.bz2  hosts.gz
bigfile2 bigfile4  bigfile-gz9.gz  bigfile-xz9.xz  gzip.tar.gz    xz.tar.xz
$ tar cf plain.tar bigfile bigfile2 bigfile3
$ tar tf plain.tar
bigfile
bigfile2
```

```
bigfile3
$ tar uf plain.tar bigfile4
$ tar tf plain.tar
bigfile
bigfile2
bigfile3
bigfile4
$ tar uzf gzip.tar.gz bigfile4
tar: Cannot update compressed archives
Try 'tar --help' or 'tar --usage' for more information.
```

Quản lý tập ZIP

Các máy Windows thường không có các ứng dụng để xử lý bóng tar hoặc nhiều công cụ nên thường thấy trên hệ thống Linux. Nếu bạn cần tương tác với các hệ thống Windows, bạn có thể sử dụng các tập ZIP. Tập ZIP là tập lưu trữ tương tự như tập tar đã nén.

Các chương trình `zip` và `unzip` có thể được sử dụng để làm việc với các tập ZIP trên hệ thống Linux. Ví dụ dưới đây là tất cả những gì bạn cần để bắt đầu sử dụng chúng. Trước tiên, chúng ta sẽ tạo một tập hợp các tập:

```
$ cd ~/linux_essentials-3.1
$ mkdir zip
$ cd zip/
$ mkdir dir
$ touch dir/file1 dir/file2
```

Bây giờ chúng ta sử dụng `zip` để đóng gói các tập này thành tập ZIP:

```
$ zip -r zipfile.zip dir
adding: dir/ (stored 0%)
adding: dir/file1 (stored 0%)
adding: dir/file2 (stored 0%)
$ rm -rf dir
```

Cuối cùng, ta giải nén lại tập ZIP:

```
$ ls
zipfile.zip
$ unzip zipfile.zip
```

```
Archive: zipfile.zip
  creating: dir/
  extracting: dir/file1
  extracting: dir/file2
$ find
.
./zipfile.zip
./dir
./dir/file1
./dir/file2
```

Khi thêm thư mục vào tệp ZIP, tùy chọn `-r` sẽ khiến `zip` bao gồm cả nội dung của thư mục. Không có nó, bạn sẽ có một thư mục trống trong tệp ZIP.

Bài tập Hướng dẫn

1. Hãy xét phần mở rộng và cho biết công cụ nào sau đây được sử dụng để tạo các tệp này?

Tên Tệp	tar	gzip	bzip2	xz
archive.tar				
archive.tgz				
archive.tar.xz				

2. Xét các phần mở rộng, tệp nào trong số này là tệp lưu trữ và tệp nào là tệp nén?

Tên tệp	Tệp Lưu trữ	Tệp Nén
file.tar		
file.tar.bz2		
file.zip		
file.xz		

3. Làm cách nào để có thể thêm tệp vào tệp tar đã nén bằng gzip?

4. Tùy chọn nào của lệnh `tar` sẽ hướng dẫn tar bao gồm cả / đứng đầu trong các đường dẫn tuyệt đối?

5. zip có hỗ trợ các mức nén khác nhau không?

Bài tập Mở rộng

1. Khi giải nén tệp, `tar` có hỗ trợ khối khớp mã trong danh sách tệp không?

2. Làm cách nào để có thể đảm bảo tệp được giải nén sẽ giống với tệp trước khi được nén?

3. Điều gì sẽ xảy ra nếu bạn cố trích xuất một tệp từ kho lưu trữ `tar` đã tồn tại trên hệ thống tệp của bạn?

4. Làm cách nào để có thể giải nén tệp `archive.tgz` mà không sử dụng tùy chọn `tar z`?

Tóm tắt

Các hệ thống Linux có sẵn một số công cụ nén và lưu trữ. Bài học này đã cho chúng ta biết về những công cụ phổ biến nhất. Công cụ lưu trữ phổ biến nhất là `tar`. Nếu cần tương tác với các hệ thống Windows, `zip` và `unzip` có thể tạo và giải nén các tập ZIP.

Lệnh `tar` có một vài tùy chọn đáng để ghi nhớ. Chúng là `x` để giải nén, `c` để tạo, `t` để xem nội dung và `u` để thêm hoặc thay thế tập. Tùy chọn `v` sẽ liệt kê các tập được `tar` xử lý trong khi tạo hoặc giải nén tập lưu trữ.

Kho lưu trữ của bản phân phối Linux điển hình có nhiều công cụ nén. Phổ biến nhất là `gzip`, `bzip2` và `xz`. Các thuật toán nén thường hỗ trợ các cấp độ nén khác nhau cho phép bạn tối ưu hóa tốc độ hoặc kích thước tập. Các tập có thể được giải nén bằng `gunzip`, `bunzip2` và `unxz`.

Các công cụ nén thường có các chương trình hoạt động giống như các công cụ tập văn bản phổ biến với sự khác biệt là chúng hoạt động trên các tập nén. Một vài trong số đó là `zcat`, `bzcat` và `xzcat`. Các công cụ nén thường đi kèm với các chương trình có chức năng của `grep`, `more`, `less`, `diff` và `cmp`.

Các lệnh được dùng trong bài học này:

bunzip2

Giải nén tập nén `bzip2`.

bzcat

Xuất nội dung của tập nén `bzip`.

bzip2

Nén tập bằng thuật toán và định dạng `bzip2`.

gunzip

Giải nén tập nén `gzip`.

gzip

Nén tập bằng thuật toán và định dạng `gzip`.

tar

Tạo, cập nhật, liệt kê và trích xuất các kho lưu trữ `tar`.

unxz

Giải nén tập nén `xz`.

unzip

Giải nén và trích xuất nội dung từ tệp ZIP.

xz Nén tệp bằng thuật toán và định dạng **xz**.

zcat

Xuất nội dung của tệp nén **gzip**.

zip

Tạo và nén các tệp lưu trữ ZIP.

Đáp án Bài tập Hướng dẫn

1. Xét các phần mở rộng, tệp nào trong số này là tệp lưu trữ và tệp nào là tệp nén?

Tên tệp	Tệp Lưu trữ	Tệp Nén
archive.tar	X	
		archive.tgz
X	X	
	archive.tar.xz	X
		X

2. Xét các phần mở rộng, tệp nào trong số này là tệp lưu trữ và tệp nào là tệp nén?

Tên tệp	Tệp Lưu trữ	Tệp Nén
file.tar	X	
file.tar.bz2	X	X
file.zip	X	X
file.xz		X

3. Làm cách nào để có thể thêm tệp vào tệp tar đã nén bằng gzip?

Giải nén tệp bằng `gunzip`, thêm tệp bằng `tar uf`, sau đó nén tệp bằng `gzip`.

4. Tùy chọn nào của lệnh `tar` sẽ hướng dẫn `tar` bao gồm cả / đứng đầu trong các đường dẫn tuyệt đối?

Tùy chọn `-P`. Từ trang hướng dẫn:

```
-P, --absolute-names
    Don't strip leading slashes from file names when creating archives
```

5. zip có hỗ trợ các mức nén khác nhau không?

Có. Sử dụng `-#`, thay thế `#` bằng một số từ 0-9. Từ trang hướng dẫn:

```
-#
(-0, -1, -2, -3, -4, -5, -6, -7, -8, -9)
```

Regulate the speed of compression using the specified digit #, where -0 indicates no compression (store all files), -1 indicates the fastest compression speed (less compression) and -9 indicates the slowest compression speed (optimal compression, ignores the suffix list). The default compression level is -6.

Though still being worked, the intention is this setting will control compression speed for all compression methods. Currently only deflation is controlled.

Đáp án Bài tập Mở rộng

1. Khi giải nén tệp, `tar` có hỗ trợ khối khớp mã trong danh sách tệp không?

Có, bằng cách sử dụng tùy chọn `--wildcards`. `--wildcards` phải được đặt ngay sau tệp `tar` khi sử dụng kiểu tùy chọn không có dấu gạch ngang. Ví dụ:

```
$ tar xf tarfile.tar --wildcards dir/file*
$ tar --wildcards -xf tarfile.tar dir/file*
```

2. Làm cách nào để có thể đảm bảo tệp được giải nén sẽ giống với tệp trước khi được nén?

Bạn không cần phải làm bất cứ điều gì khi sử dụng các công cụ được đề cập đến trong bài học này. Cả ba đều bao gồm tổng kiểm tra ở định dạng tệp của chúng được xác minh khi chúng được giải nén.

3. Điều gì sẽ xảy ra nếu bạn cố trích xuất một tệp từ kho lưu trữ `tar` đã tồn tại trên hệ thống tệp của bạn?

Tệp trên hệ thống tệp của bạn sẽ bị ghi đè bằng phiên bản có trong tệp `tar`.

4. Làm cách nào để có thể giải nén tệp `archive.tgz` mà không sử dụng tùy chọn `tar z`?

Trước tiên, bạn sẽ giải nén nó bằng `gunzip`.

```
$ gunzip archive.tgz
$ tar xf archive.tar
```



3.2 Tìm kiếm và trích xuất dữ liệu từ Tập

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 3.2](#)

Khối lượng

3

Các lĩnh vực kiến thức chính

- Đường ống dòng lệnh
- Chuyển hướng vào/ra
- Biểu thức chính quy cơ bản sử dụng `.`, `[]`, `*`, và `?`

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

- `grep`
- `less`
- `cat`, `head`, `tail`
- `sort`
- `cut`
- `wc`



3.2 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	3 Sức mạnh của Dòng lệnh
Mục tiêu:	3.2 Tìm kiếm và Trích xuất Dữ liệu từ Tập
Bài:	1 trên 2

Giới thiệu

Trong bài học này, chúng ta sẽ tập trung vào việc chuyển hướng hoặc truyền thông tin từ nguồn này sang nguồn khác với sự trợ giúp của các công cụ cụ thể. Dòng lệnh Linux chuyển hướng thông tin qua các kênh tiêu chuẩn cụ thể. Đầu vào tiêu chuẩn (*stdin* hoặc kênh 0) của lệnh được coi là bàn phím và đầu ra tiêu chuẩn (*stdout* hoặc kênh 1) được coi là màn hình. Ngoài ra còn có một kênh khác được sử dụng để chuyển hướng đầu ra lỗi (*stderr* hoặc kênh 2) của lệnh hoặc thông báo lỗi của chương trình. Đầu vào và/hoặc đầu ra có thể được chuyển hướng.

Khi chạy một lệnh, đôi khi chúng ta sẽ muốn truyền một số thông tin đến lệnh hoặc chuyển hướng đầu ra đến một tệp cụ thể. Từng chức năng này sẽ được giới thiệu trong hai phần tiếp theo.

Chuyển hướng I/O

Chuyển hướng I/O cho phép người dùng chuyển hướng thông tin từ hoặc tới một lệnh bằng cách sử dụng tệp văn bản. Như đã mô tả trước ở trên, đầu vào, đầu ra tiêu chuẩn và đầu ra lỗi có thể được chuyển hướng và thông tin có thể được lấy từ các tệp văn bản.

Chuyển hướng Đầu ra Tiêu chuẩn

Để chuyển hướng đầu ra tiêu chuẩn sang một tệp thay vì màn hình, chúng ta cần sử dụng toán tử `>` kèm theo sau là tên của tệp. Nếu tệp không tồn tại thì một tệp mới sẽ được tạo; nếu tệp đã có sẵn thì thông tin sẽ ghi đè lên tệp này.

Để xem nội dung của tệp vừa tạo, chúng ta có thể sử dụng lệnh `cat`. Theo mặc định, lệnh này sẽ hiển thị nội dung của một tệp trên màn hình. Hãy tham khảo trang hướng dẫn để tìm hiểu thêm về các chức năng của nó.

Ví dụ dưới đây sẽ minh họa chức năng của toán tử. Trong trường hợp đầu tiên, một tệp mới được tạo có chứa văn bản “Hello World!”:

```
$ echo "Hello World!" > text
$ cat text
Hello World!
```

Trong lần gọi thứ hai, tệp đó sẽ bị ghi đè bằng văn bản mới:

```
$ echo "Hello!" > text
$ cat text
Hello!
```

Nếu muốn thêm thông tin mới vào cuối tệp, chúng ta cần sử dụng toán tử `>>`. Toán tử này cũng sẽ tạo một tệp mới nếu nó không thể tìm thấy tệp hiện có.

Ví dụ đầu tiên cho ta thấy về việc bổ sung văn bản. Như có thể thấy, văn bản mới đã được thêm vào dòng sau:

```
$ echo "Hello to you too!" >> text
$ cat text
Hello!
Hello to you too!
```

Ví dụ thứ hai minh họa việc một tệp mới sẽ được tạo:

```
$ echo "Hello to you too!" >> text2
$ cat text2
Hello to you too!
```

Chuyển hướng Lỗi Tiêu chuẩn

Để chỉ chuyển hướng các thông báo lỗi, người dùng sẽ cần sử dụng toán tử `2>` kèm theo sau là tên của tệp chứa lỗi. Nếu tệp không tồn tại thì tệp mới sẽ được tạo; nếu tệp đã có sẵn thì nó sẽ bị ghi đè.

Như đã giải thích, kênh chuyển hướng lỗi tiêu chuẩn là *kênh 2*. Kênh phải được chỉ định khi chuyển hướng lỗi tiêu chuẩn, trái ngược với đầu ra tiêu chuẩn khác nơi *kênh 1* được chỉ định theo mặc định. Ví dụ: lệnh sau sẽ tìm kiếm tệp hoặc thư mục có tên `games` và chỉ ghi lỗi vào tệp `text-error`, cùng lúc hiển thị đầu ra tiêu chuẩn trên màn hình:

```
$ find /usr games 2> text-error
/usr
/usr/share
/usr/share/misc
-----Omitted output-----
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
$ cat text-error
find: `games': No such file or directory
```

NOTE Để biết thêm thông tin về lệnh `find`, hãy tham khảo trang hướng dẫn của nó.

Ví dụ: lệnh sau sẽ chạy mà không có lỗi, do đó sẽ không có thông tin nào được ghi lại trong tệp `text-error`:

```
$ sort /etc/passwd 2> text-error
$ cat text-error
```

Cũng như đầu ra tiêu chuẩn, lỗi tiêu chuẩn cũng có thể được thêm vào một tệp có toán tử `2>>`. Thao tác này sẽ thêm lỗi mới vào cuối tệp. Nếu tệp không tồn tại, một tệp mới sẽ được tạo. Ví dụ đầu tiên cho thấy việc bổ sung thông tin mới vào tệp, trong khi ví dụ thứ hai cho thấy lệnh tạo một tệp mới khi không thể tìm thấy tệp có cùng tên hiện có:

```
$ sort /etc 2>> text-error
$ cat text-error
sort: read failed: /etc: Is a directory
```

```
$ sort /etc/shadow 2>> text-error2
```

```
$ cat text-error2
sort: open failed: /etc/shadow: Permission denied
```

Khi sử dụng loại chuyển hướng này, sẽ chỉ có các thông báo lỗi được chuyển hướng đến tệp, đầu ra bình thường sẽ được ghi trên màn hình hoặc chuyển qua đầu ra tiêu chuẩn hoặc *stdout*.

Có một tệp cụ thể mà về mặt kỹ thuật có thể coi là một *bit bucket* (một tệp chấp nhận đầu vào và không thực hiện bất kỳ hành động nào với nó): `/dev/null`. Bạn có thể chuyển hướng bất kỳ thông tin không liên quan nào mà bạn không muốn được hiển thị hoặc được chuyển hướng vào một tệp quan trọng như minh họa trong ví dụ dưới đây:

```
$ sort /etc 2> /dev/null
```

Chuyển hướng Đầu vào Tiêu chuẩn

Loại chuyển hướng này được sử dụng để nhập dữ liệu cho một lệnh từ một tệp được chỉ định thay vì bàn phím. Trong trường hợp này, toán tử `<` được sử dụng như minh họa trong ví dụ dưới đây:

```
$ cat < text
Hello!
Hello to you too!
```

Chuyển hướng đầu vào tiêu chuẩn thường được sử dụng với các lệnh không chấp nhận đối số tệp. Lệnh `tr` là một trong số đó. Lệnh này có thể được sử dụng để biên dịch nội dung tệp bằng cách sửa đổi các ký tự trong tệp theo những cách cụ thể, chẳng hạn như xóa bất kỳ ký tự cụ thể nào khỏi tệp; ví dụ bên dưới cho thấy việc xóa ký tự `l`:

```
$ tr -d "l" < text
Heo!
Heo to you too!
```

Để biết thêm thông tin, hãy tham khảo trang hướng dẫn của `tr`.

Here Documents

Không giống như việc chuyển hướng đầu ra, toán tử `<<` hoạt động theo một cách khác so với các toán tử khác. Luồng đầu vào này còn được gọi là *here document*. *Here document* đại diện cho khối mã hoặc văn bản có thể được chuyển hướng đến lệnh hoặc chương trình tương tác. Các loại ngôn ngữ kịch bản khác nhau như `bash`, `sh` và `csh` có thể lấy đầu vào trực tiếp từ dòng lệnh mà không

cần sử dụng bất kỳ tệp văn bản nào.

Như có thể thấy trong ví dụ dưới đây, toán tử được sử dụng để nhập dữ liệu vào lệnh, trong khi từ đứng sau lại không chỉ định tên tệp. Từ này được hiểu là dấu phân tách của đầu vào và nó sẽ không được coi là một phần của nội dung, do đó mà `cat` sẽ không hiển thị nó:

```
$ cat << hello
> hey
> ola
> hello
hey
ola
```

Tham khảo trang hướng dẫn của lệnh `cat` để tìm thêm thông tin.

Kết hợp

Sự kết hợp đầu tiên mà chúng ta sẽ khám phá là giữa việc chuyển hướng đầu ra tiêu chuẩn và đầu ra lỗi tiêu chuẩn vào cùng một tệp. Các toán tử `&>` và `&>>` sẽ được sử dụng; `&` biểu thị sự kết hợp của *kênh 1* và *kênh 2*. Toán tử đầu tiên sẽ ghi đè lên nội dung hiện có của tệp và toán tử thứ hai sẽ nối hoặc thêm thông tin mới vào cuối tệp. Cả hai toán tử sẽ cho phép tạo tệp mới nếu nó chưa tồn tại giống như ở trong các phần trước:

```
$ find /usr admin &> newfile
$ cat newfile
/usr
/usr/share
/usr/share/misc
-----Omitted output-----
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
find: `admin': No such file or directory
$ find /etc/calendar &>> newfile
$ cat newfile
/usr
/usr/share
/usr/share/misc
-----Omitted output-----
/usr/lib/libmagic.so.1.0.0
/usr/lib/libdns.so.81
/usr/games
```

```
find: `admin': No such file or directory
/etc/calendar
/etc/calendar/default
```

Hãy xem một ví dụ sử dụng lệnh `cut`:

```
$ cut -f 3 -d "/" newfile
$ cat newfile

share
share
share
-----Omitted output-----
lib
games
find: `admin': No such file or directory
calendar
calendar
find: `admin': No such file or directory
```

Lệnh `cut` sẽ cắt các trường được chỉ định khỏi tệp đầu vào bằng cách sử dụng tùy chọn `-f`, trong trường hợp này sẽ là trường thứ 3. Để lệnh tìm được trường, dấu phân tách cũng cần phải được chỉ định bằng tùy chọn `-d`. Trong trường hợp của chúng ta, ký tự phân tách sẽ là dấu `/`.

Để tìm hiểu thêm về lệnh `cut`, hãy tham khảo trang hướng dẫn của nó.

Đường ống Dòng lệnh

Việc chuyển hướng chủ yếu được sử dụng để lưu trữ kết quả của một lệnh sẽ được xử lý bởi một lệnh khác. Loại quy trình trung gian này có thể trở nên rất mất công và phức tạp nếu bạn muốn dữ liệu trải qua nhiều quy trình. Để tránh điều này, bạn có thể liên kết lệnh trực tiếp qua *các đường ống*. Nói cách khác, đầu ra của lệnh đầu tiên sẽ tự động trở thành đầu vào của lệnh thứ hai. Kết nối này được thực hiện bằng cách sử dụng toán tử `|` (thanh dọc):

```
$ cat /etc/passwd | less
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
:
```

Trong ví dụ trên, lệnh `less` đứng sau toán tử đường ống sẽ sửa đổi cách hiển thị tệp. Lệnh `less`

hiển thị tệp văn bản cho phép người dùng cuộn lên và xuống một dòng tại thời điểm đó. `less` cũng được sử dụng theo mặc định để hiển thị các trang hướng dẫn như đã được nhắc tới trong các bài học trước.

Có thể sử dụng nhiều đường ống cùng một lúc. Các lệnh trung gian sẽ nhận đầu vào, sau đó thay đổi nó và tạo ra đầu ra được gọi là *bộ lọc*. Hãy sử dụng lệnh `ls -l` và thử đếm số từ trong 10 dòng đầu tiên của kết quả. Để làm điều này, chúng ta sẽ phải sử dụng lệnh `head` mà theo mặc định sẽ hiển thị 10 dòng đầu tiên của tệp và sau đó đếm số từ bằng lệnh `wc`:

```
$ ls -l | head | wc -w
10
```

Như đã đề cập trước đó, theo mặc định, `head` sẽ chỉ hiển thị 10 dòng đầu tiên của tệp văn bản được chỉ định. Hành vi này có thể được sửa đổi bằng cách sử dụng các tùy chọn cụ thể. Hãy kiểm tra trang hướng dẫn của lệnh để tìm thêm thông tin.

Như đã đề cập trước đó, theo mặc định, `head` chỉ hiển thị 10 dòng đầu tiên của tệp văn bản được chỉ định. Hành vi này có thể được sửa đổi bằng cách sử dụng các tùy chọn cụ thể. Hãy kiểm tra trang hướng dẫn của lệnh để tìm thêm thông tin.

NOTE

Tùy chọn `-f` có thể hiển thị các dòng cuối cùng của tệp trong khi tệp đang được cập nhật. Tính năng này có thể trở nên rất hữu ích khi giám sát một tệp như `syslog` cho hoạt động đang diễn ra.

Theo mặc định, lệnh `wc` (word count - số từ) sẽ đếm số dòng, từ và byte của tệp. Như được nhắc đến trong bài tập, tùy chọn `-w` sẽ khiến lệnh chỉ đếm từ ở trong các dòng đã chọn. Các tùy chọn phổ biến nhất mà bạn có thể sử dụng với lệnh này là `-l` để chỉ định cho lệnh chỉ đếm các dòng và `-c` được sử dụng để chỉ đếm các byte. Có thể tìm thấy trong trang hướng dẫn của lệnh nhiều biến thể và tùy chọn khác của lệnh cũng như thêm thông tin về `wc`.

Bài tập Hướng dẫn

1. Hãy liệt kê nội dung của thư mục hiện tại của bạn, bao gồm quyền sở hữu và các quyền, đồng thời chuyển hướng đầu ra tới một tệp có tên `contents.txt` trong thư mục chính của bạn.

2. Hãy sắp xếp nội dung của tệp `contents.txt` từ thư mục hiện tại của bạn và nối nó vào cuối tệp mới có tên `contents-sorted.txt`.

3. Hãy hiển thị 10 dòng cuối cùng của tệp `/etc/passwd` và chuyển hướng nó tới một tệp mới trong thư mục Documents người dùng của bạn.

4. Hãy đếm số từ trong tệp `contents.txt` và nối đầu ra vào cuối tệp `field2.txt` trong thư mục chính của bạn. Bạn sẽ cần sử dụng cả chuyển hướng đầu vào và đầu ra.

5. Hãy hiển thị 5 dòng đầu tiên của tệp `/etc/passwd` và sắp xếp đầu ra theo thứ tự bảng chữ cái đảo ngược.

6. Sử dụng tệp `contents.txt` đã tạo trước đó, hãy đếm số ký tự của 9 dòng cuối cùng.

7. Hãy đếm số lượng tệp có tên `test` trong thư mục `/usr/share` và các thư mục con của nó. Lưu ý: mỗi dòng đầu ra từ lệnh `find` sẽ đại diện cho một tệp.

Bài tập Mở rộng

1. Hãy chọn trường thứ hai của tập `contents.txt` và chuyển hướng đầu ra tiêu chuẩn và đầu ra lỗi sang một tập khác có tên `field1.txt`.

2. Sử dụng toán tử chuyển hướng đầu vào và lệnh `tr`, hãy xóa dấu gạch ngang (-) khỏi tập `contents.txt`.

3. Lợi ích lớn nhất của việc chỉ chuyển hướng lỗi đến một tập là gì?

4. Hãy thay thế tất cả các khoảng trắng lặp lại trong tập `contents.txt` được sắp xếp theo thứ tự bảng chữ cái bằng một khoảng trắng.

5. Trong một dòng lệnh, hãy loại bỏ các khoảng trắng lặp lại (như đã thực hiện trong bài tập trước), chọn trường thứ chín và sắp xếp theo thứ tự bảng chữ cái đảo ngược và không phân biệt chữ hoa chữ thường. Bạn đã phải sử dụng bao nhiêu đường ống?

Tóm tắt

Trong bài học này, bạn đã học về:

- Các loại chuyển hướng
- Cách sử dụng các toán tử chuyển hướng
- Cách sử dụng đường ống để lọc đầu ra của lệnh

Các lệnh được sử dụng trong bài này:

cut

Loại bỏ các phần từ mỗi dòng của một tệp.

cat

Hiển thị hoặc nối các tệp.

find

Tìm kiếm các tệp trong một hệ thống phân cấp thư mục.

less

Hiển thị một tệp, cho phép người dùng cuộn một dòng tại một thời điểm.

more

Hiển thị một tệp, một trang tại thời điểm đó.

head

Hiển thị 10 dòng đầu tiên của tệp.

tail

Hiển thị 10 dòng cuối cùng của tệp.

sort

Sắp xếp các tệp.

wc

Đếm theo mặc định số dòng, từ hoặc byte của tệp.

Đáp án Bài tập Hướng dẫn

1. Hãy liệt kê nội dung của thư mục hiện tại của bạn, bao gồm quyền sở hữu và các quyền, đồng thời chuyển hướng đầu ra tới một tệp có tên `contents.txt` trong thư mục chính của bạn.

```
$ ls -l > contents.txt
```

2. Hãy sắp xếp nội dung của tệp `contents.txt` từ thư mục hiện tại của bạn và nối nó vào cuối tệp mới có tên `contents-sorted.txt`.

```
$ sort contents.txt >> contents-sorted.txt
```

3. Hãy hiển thị 10 dòng cuối cùng của tệp `/etc/passwd` và chuyển hướng nó tới một tệp mới trong thư mục Documents người dùng của bạn.

```
$ tail /etc/passwd > Documents/newfile
```

4. Hãy đếm số từ trong tệp `contents.txt` và nối đầu ra vào cuối tệp `field2.txt` trong thư mục chính của bạn. Bạn sẽ cần sử dụng cả chuyển hướng đầu vào và đầu ra.

```
$ wc < contents.txt >> field2.txt
```

5. Hãy hiển thị 5 dòng đầu tiên của tệp `/etc/passwd` và sắp xếp đầu ra theo thứ tự bảng chữ cái đảo ngược.

```
$ head -n 5 /etc/passwd | sort -r
```

6. Sử dụng tệp `contents.txt` đã tạo trước đó, hãy đếm số ký tự của 9 dòng cuối cùng.

```
$ tail -n 9 contents.txt | wc -c  
531
```

7. Hãy đếm số lượng tệp có tên `test` trong thư mục `/usr/share` và các thư mục con của nó. Lưu ý: mỗi dòng đầu ra từ lệnh `find` sẽ đại diện cho một tệp.

```
$ find /usr/share -name test | wc -l
```

125

Đáp án Bài tập Mở rộng

1. Hãy chọn trường thứ hai của tệp `contents.txt` và chuyển hướng đầu ra tiêu chuẩn và đầu ra lỗi sang một tệp khác có tên `field1.txt`.

```
$ cut -f 2 -d " " contents.txt > field1.txt
```

2. Sử dụng toán tử chuyển hướng đầu vào và lệnh `tr`, hãy xóa dấu gạch ngang (-) khỏi tệp `contents.txt`.

```
$ tr -d "-" < contents.txt
```

3. Lợi ích lớn nhất của việc chỉ chuyển hướng lỗi đến một tệp là gì?

Chỉ chuyển hướng lỗi đến một tệp có thể giúp giữ tệp nhật ký được theo dõi thường xuyên.

4. Hãy thay thế tất cả các khoảng trắng lặp lại trong tệp `contents.txt` được sắp xếp theo thứ tự bảng chữ cái bằng một khoảng trắng.

```
$ sort contents.txt | tr -s " "
```

5. Trong một dòng lệnh, hãy loại bỏ các khoảng trắng lặp lại (như đã thực hiện trong bài tập trước), chọn trường thứ chín và sắp xếp theo thứ tự bảng chữ cái đảo ngược và không phân biệt chữ hoa chữ thường. Bạn đã phải sử dụng bao nhiêu đường ống?

```
$ cat contents.txt | tr -s " " | cut -f 9 -d " " | sort -fr
```

Bài tập sử dụng 3 ống, mỗi ống dành cho một bộ lọc.



3.2 Bài 2

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	3 Sức mạnh của Dòng lệnh
Mục tiêu:	3.2 Tìm kiếm và Trích xuất Dữ liệu từ Tập
Bài:	2 trên 2

Giới thiệu

Trong bài học này, chúng ta sẽ tìm hiểu về các công cụ được sử dụng để thao tác với văn bản. Những công cụ này thường được quản trị viên hệ thống hoặc chương trình sử dụng để tự động theo dõi hoặc xác định thông tin định kỳ cụ thể.

Tìm kiếm trong Tập bằng grep

Công cụ đầu tiên mà chúng ta sẽ thảo luận trong bài học này là lệnh `grep`. `grep` là chữ viết tắt của cụm từ “global regular expression print” (in biểu thức chính quy toàn cầu) và chức năng chính của nó là tìm kiếm mẫu đã được chỉ định trong các tập. Lệnh này sẽ xuất dòng chứa mẫu đã được chỉ định và đánh dấu mẫu bằng màu đỏ.

```
$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
user:x:1001:1001:User,,,:/home/user:/bin/bash
```

Giống như hầu hết các lệnh, `grep` cũng có thể được tinh chỉnh bằng cách sử dụng các tùy chọn.

Dưới đây là những tùy chọn phổ biến nhất:

-i

tìm kiếm không phân biệt chữ hoa chữ thường

-r

tìm kiếm là đệ quy (tìm kiếm tất cả các tệp trong thư mục đã được chỉ định và các thư mục con của nó)

-c

tìm kiếm đếm số kết quả khớp

-v

đảo ngược kết quả khớp để in các dòng không khớp với từ khoá tìm kiếm

-E

bật các biểu thức chính quy mở rộng (cần thiết cho một số ký tự meta nâng cao hơn như `|`, `+` và `?`)

grep còn có nhiều tùy chọn hữu ích khác. Hãy tham khảo trang hướng dẫn để tìm hiểu thêm về nó.

Biểu thức Chính quy

Công cụ thứ hai là một công cụ rất mạnh mẽ. Nó được sử dụng để mô tả các bit văn bản trong tệp và còn được gọi là *biểu thức chính quy*. Biểu thức chính quy cực kỳ hữu ích trong việc trích xuất dữ liệu từ tệp văn bản bằng cách xây dựng các mẫu. Chúng thường được sử dụng trong các tệp lệnh hoặc khi lập trình với các ngôn ngữ cấp cao, chẳng hạn như Perl hoặc Python.

Khi làm việc với các biểu thức chính quy, điều rất quan trọng cần lưu ý là *mọi ký tự đều được tính*, và mẫu được viết với mục đích khớp với một chuỗi ký tự cụ thể, hay còn gọi là chuỗi. Hầu hết các mẫu sử dụng các ký tự ASCII thông thường, chẳng hạn như chữ cái, chữ số, dấu chấm câu hoặc các ký hiệu khác, nhưng nó cũng có thể sử dụng các ký tự Unicode để khớp với bất kỳ loại văn bản nào khác.

Danh sách sau đây sẽ giải thích các ký tự meta của biểu thức chính quy được sử dụng để tạo các mẫu.

.

So khớp với bất kỳ ký tự đơn nào (ngoại trừ ký tự xuống dòng)

[abcABC]

So khớp bất kỳ một ký tự nào trong dấu ngoặc

[^abcABC]

So khớp với bất kỳ ký tự nào ngoại trừ ký tự trong ngoặc

[a-z]

So khớp với bất kỳ ký tự nào trong phạm vi

[^a-z]

So khớp với bất kỳ ký tự nào ngoại trừ các ký tự trong phạm vi

sun|moon

Tìm một trong các chuỗi được liệt kê

^

Bắt đầu một dòng

\$

Kết thúc dòng

Tất cả các chức năng của biểu thức chính quy cũng có thể được triển khai thông qua `grep`. Bạn có thể thấy rằng trong ví dụ trên, từ này không được bao quanh bởi dấu trích dẫn kép. Để ngăn vỡ diễn giải siêu ký tự, ta nên sử dụng một mẫu phức tạp hơn ở giữa dấu trích dẫn kép (" "). Với mục đích thực hành, chúng ta sẽ sử dụng dấu trích dẫn kép khi triển khai các biểu thức chính quy. Các dấu trích dẫn kép khác sẽ giữ nguyên chức năng thông thường của chúng như đã được thảo luận trong các bài học trước.

Các ví dụ sau đây sẽ nhấn mạnh chức năng của các biểu thức chính quy. Chúng ta sẽ cần dữ liệu trong tệp; do đó, nhóm lệnh tiếp theo sẽ chỉ nối các chuỗi khác nhau vào tệp `text.txt`.

```
$ echo "aaabbb1" > text.txt
$ echo "abab2" >> text.txt
$ echo "noone2" >> text.txt
$ echo "class1" >> text.txt
$ echo "alien2" >> text.txt
$ cat text.txt
aaabbb1
abab2
noone2
class1
```

```
alien2
```

Ví dụ đầu tiên là sự kết hợp tìm kiếm thông qua tệp có và không có biểu thức chính quy. Để hiểu rõ về các biểu thức chính quy, điều quan trọng là chỉ ra sự khác biệt. Lệnh đầu tiên tìm kiếm chính xác chuỗi ở bất kỳ vị trí nào trong dòng, trong khi lệnh thứ hai lại tìm kiếm các bộ ký tự chứa bất kỳ ký tự nào nằm giữa các dấu trích dẫn. Do đó, kết quả của các lệnh sẽ khác nhau.

```
$ grep "ab" text.txt
aaabbb1
abab2
$ grep "[ab]" text.txt
aaabbb1
abab2
class1
alien2
```

Nhóm ví dụ thứ hai cho thấy ứng dụng của ký tự meta ở đầu và cuối dòng. Điều quan trọng là cần phải đặt 2 ký tự ở đúng vị trí trong biểu thức. Khi chỉ định đầu dòng, siêu ký tự cần phải đứng trước biểu thức; khi chỉ định cuối dòng, siêu ký tự cần phải đứng sau biểu thức.

```
$ grep "^a" text.txt
aaabbb1
abab2
alien2
$ grep "2$" text.txt
abab2
noone2
alien2
```

Ngoài các ký tự meta được giải thích trước đó, các biểu thức chính quy cũng có các ký tự meta cho phép nhân mẫu đã chỉ định trước đó:

*

Không hoặc nhiều mẫu trước đó

+

Một hoặc nhiều mẫu trước đó

?

Không hoặc một trong các mẫu trước đó

Đối với các ký tự meta cấp số nhân, lệnh bên dưới sẽ tìm kiếm một chuỗi chứa `ab`, một ký tự đơn và một hoặc nhiều ký tự đã tìm thấy trước đó. Kết quả cho thấy `grep` đã tìm thấy chuỗi `aaabbb1` khớp với phần `abbb` cũng như `abab2`. Vì ký tự `+` là ký tự biểu thức chính quy *mở rộng* nên chúng ta cần chuyển tùy chọn `-E` cho lệnh `grep`.

```
$ grep -E "ab.+" text.txt
aaabbb1
abab2
```

Hầu hết các ký tự meta đều khá dễ hiểu; nhưng trong những lần sử dụng đầu tiên thì chúng lại có thể trở nên khá phức tạp. Các ví dụ ở trên đại diện cho một phần nhỏ chức năng của biểu thức chính quy. Hãy thử tất cả các siêu ký tự từ bảng trên để hiểu thêm về cách chúng hoạt động.

Bài tập Hướng dẫn

Sử dụng `grep` và tệp `/usr/share/hunspell/en_US.dic`, hãy tìm các dòng phù hợp với các tiêu chí sau:

1. Tất cả các dòng chứa từ `cat` ở bất kỳ vị trí nào trên dòng.

2. Tất cả các dòng không chứa bất kỳ ký tự nào sau đây: `sawgtfixk`.

3. Tất cả các dòng bắt đầu bằng 3 chữ cái bất kỳ và từ `dig`.

4. Tất cả các dòng kết thúc bằng ít nhất một chữ ``e``.

5. Tất cả các dòng chứa một trong các từ sau: `org`, `kay` hoặc `tuna`.

6. Số dòng bắt đầu bằng một hoặc không chữ `c`, theo sau là chuỗi `ati`.

Bài tập Mở rộng

1. Hãy tìm biểu thức chính quy khớp với các từ trong dòng “Bao gồm” và không khớp với các từ trong dòng “Loại trừ”:

- Bao gồm: pot, spot, apot

Loại trừ: potic, spots, potatoe

- Bao gồm: arp99, apple, zipper

Loại trừ: zoo, arive, attack

- Bao gồm: arcane, capper, zoology

Loại trừ: air, coper, zoloc

- Bao gồm: 0th/pt, 3th/tc, 9th/pt

Loại trừ: 0/nm, 3/nm, 9/nm

- Bao gồm: Hawaii, Dario, Ramiro

Loại trừ: hawaii, Ian, Alice

2. Có những lệnh nào khác hữu ích thường được sử dụng để tìm kiếm trong các tệp không? Nó có những chức năng bổ sung nào?

3. Hãy nhớ lại bài học trước và sử dụng một trong các ví dụ để thử tìm kiếm một mẫu cụ thể trong đầu ra của lệnh với sự trợ giúp của `grep`.

Tóm tắt

Trong bài học này, bạn đã học về:

- Biểu thức chính quy siêu ký tự
- Cách tạo mẫu với biểu thức chính quy
- Cách tìm kiếm trong các tệp

Các lệnh được dùng trong bài tập:

grep

Tìm kiếm các ký tự hoặc chuỗi trong một tệp

Đáp án Bài tập Hướng dẫn

Sử dụng `grep` và tệp `/usr/share/hunspell/en_US.dic`, hãy tìm các dòng phù hợp với các tiêu chí sau:

1. Tất cả các dòng chứa từ `cat` ở bất kỳ vị trí nào trên dòng.

```
$ grep "cat" /usr/share/hunspell/en_US.dic
Alcatraz/M
Decatur/M
Hecate/M
...
```

2. Tất cả các dòng không chứa bất kỳ ký tự nào sau đây: `sawgtfixk`.

```
$ grep -v "[sawgtfixk]" /usr/share/hunspell/en_US.dic
49269
0/nm
1/n1
2/nm
2nd/p
3/nm
3rd/p
4/nm
5/nm
6/nm
7/nm
8/nm
...
```

3. Tất cả các dòng bắt đầu bằng 3 chữ cái bất kỳ và từ `dig`.

```
$ grep "^...dig" /usr/share/hunspell/en_US.dic
cardigan/SM
condign
predigest/GDS
...
```

4. Tất cả các dòng kết thúc bằng ít nhất một chữ `e``.

```
$ grep -E "e+$" /usr/share/hunspell/en_US.dic
Anglicize
Anglophobe
Anthropocene
...
```

5. Tất cả các dòng chứa một trong các từ sau: `org` , `kay` hoặc `tuna`.

```
$ grep -E "org|kay|tuna" /usr/share/hunspell/en_US.dic
Borg/SM
George/MS
Tokay/M
fortunate/UY
...
```

6. Số dòng bắt đầu bằng một hoặc không chữ `c`, theo sau là chuỗi `ati`.

```
$ grep -cE "^c?ati" /usr/share/hunspell/en_US.dic
3
```

Đáp án Bài tập Mở rộng

1. Hãy tìm biểu thức chính quy khớp với các từ trong dòng “Bao gồm” và không khớp với các từ trong dòng “Loại trừ”:

- Bao gồm: pot, spot, apot

Loại trừ: potic, spots, potatoe

+ Đáp án: pot\$

- Bao gồm: arp99, apple, zipper

Loại trừ: zoo, arive, attack

Đáp án: p+

- Bao gồm: arcane, capper, zoology

Loại trừ: air, coper, zoloc

Đáp án: arc|cap|zoo

- Bao gồm: 0th/pt, 3th/tc, 9th/pt

Loại trừ: 0/nm, 3/nm, 9/nm

Đáp án: [0-9]th.+

- Bao gồm: Hawaii, Dario, Ramiro

Loại trừ: hawaii, Ian, Alice

Đáp án: ^[A-Z]a.*i+

2. Có những lệnh nào khác hữu ích thường được sử dụng để tìm kiếm trong các tệp không? Nó có những chức năng bổ sung nào?

Lệnh sed. Lệnh này có thể tìm và thay thế các ký tự hoặc bộ ký tự trong một tệp.

3. Hãy nhớ lại bài học trước và sử dụng một trong các ví dụ để thử tìm kiếm một mẫu cụ thể trong đầu ra của lệnh với sự trợ giúp của grep.

Lấy một trong các câu trả lời từ Bài tập Mở rộng và tìm dòng có thể đọc, viết và thực thi dưới dạng quyền nhóm. Câu trả lời của bạn có thể khác, tùy thuộc vào lệnh bạn đã chọn và mẫu bạn

đã tạo.

```
$ cat contents.txt | tr -s " " | grep "^...rwx"
```

Bài tập này nhằm cho bạn thấy rằng `grep` cũng có thể nhận đầu vào từ các lệnh khác nhau và nó có thể giúp lọc thông tin được tạo.



3.3 Biến Lệnh thành Tập lệnh

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 3.3](#)

Khối lượng

4

Các lĩnh vực kiến thức chính

- Tập lệnh vỏ cơ bản
- Nhận biết về các trình soạn thảo văn bản phổ biến (vi và nano)

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

- `#!` (shebang)
- `/bin/bash`
- Variables
- Arguments
- `for` loops
- `echo`
- Exit status



3.3 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	3 Sức mạnh của Dòng lệnh
Mục tiêu:	3.3 Biến các Lệnh thành một Tập lệnh
Bài:	1 trên 2

Giới thiệu

Cho đến nay, chúng ta đã học cách thực thi các lệnh từ vỏ, nhưng chúng ta cũng có thể nhập các lệnh vào một tệp, sau đó đặt tệp đó thành tệp có thể thực thi được. Khi tệp được thực thi, các lệnh đó sẽ được chạy lần lượt. Các tệp thực thi này được gọi là *tệp lệnh* và chúng là một công cụ cực kỳ quan trọng đối với bất kỳ quản trị viên hệ thống Linux nào. Về cơ bản, chúng ta có thể coi Bash vừa là ngôn ngữ lập trình, vừa là vỏ.

Đầu ra In

Hãy bắt đầu bằng cách minh họa một lệnh mà bạn đã thấy trong các bài học trước: `echo` sẽ in một đối số ra đầu ra tiêu chuẩn.

```
$ echo "Hello World!"  
Hello World!
```

Bây giờ, chúng ta sẽ sử dụng chuyển hướng tệp để gửi lệnh này đến một tệp mới có tên `new_script`.

```
$ echo 'echo "Hello World!"' > new_script
$ cat new_script
echo "Hello World!"
```

Tệp `new_script` hiện đã chứa lệnh giống như trên.

Tạo Tệp lệnh Thực thi

Hãy cùng minh họa một số bước cần thiết để làm cho tệp này thực thi theo cách ta mong đợi. Nhiều khả năng người dùng sẽ nghĩ ngay đến việc chỉ cần gõ tên của tệp lệnh giống như cách họ có thể gõ tên của bất kỳ lệnh nào khác:

```
$ new_script
/bin/bash: new_script: command not found
```

Chúng ta có thể giả định rằng `new_script` có tồn tại ở vị trí hiện tại của chúng ta, nhưng phải lưu ý rằng thông báo lỗi sẽ không cho chúng ta biết rằng *tệp* không tồn tại mà nó sẽ cho chúng ta biết rằng *lệnh* không tồn tại. Sẽ rất hữu ích nếu chúng ta thảo luận về cách Linux xử lý các lệnh và các tệp thực thi.

Lệnh và PATH

Ví dụ: khi chúng ta nhập lệnh `ls` vào vỏ, ta đang thực thi một tệp có tên là `ls` tồn tại trong hệ thống tệp. Bạn có thể chứng minh điều này bằng cách sử dụng `which``:

```
$ which ls
/bin/ls
```

Nếu mỗi khi muốn xem nội dung của một thư mục ta lại phải tự tay gõ từng chữ một đường dẫn tuyệt đối của `ls` thì rất mất thời gian, vì vậy Bash có một *biến môi trường* để chứa tất cả các thư mục nơi chúng ta có thể tìm thấy các lệnh mà ta muốn chạy. Bạn có thể xem nội dung của biến này bằng cách sử dụng `echo`.

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

Mỗi vị trí này là nơi vỏ dự kiến sẽ tìm thấy một lệnh và sẽ được phân tách bằng dấu hai chấm (:).

Bạn sẽ nhận thấy rằng `/bin` cũng có mặt, nhưng có thể giả định rằng vị trí hiện tại của chúng ta thì lại không có. Vở sẽ tìm kiếm `new_script` trong mỗi thư mục này, nhưng nó sẽ không tìm được và do đó sẽ đưa ra thông báo lỗi như chúng ta đã thấy ở trên.

Có ba giải pháp cho vấn đề này: chúng ta có thể di chuyển `new_script` vào một trong các thư mục `PATH`, có thể thêm thư mục hiện tại của mình vào `PATH` hoặc thay đổi cách gọi tệp lệnh. Giải pháp thứ hai là dễ nhất, nó chỉ yêu cầu ta xác định *vị trí hiện tại* khi gọi tệp lệnh bằng cách sử dụng dấu gạch chéo (`./`).

```
$ ./new_script
/bin/bash: ./new_script: Permission denied
```

Thông báo lỗi đã thay đổi, tức là chúng ta đã đạt được một số tiến triển.

Thực thi các Quyền

Việc đầu tiên mà người dùng nên thực hiện trong trường hợp này là sử dụng `ls -l` để xem tệp:

```
$ ls -l new_script
-rw-rw-r-- 1 user user 20 Apr 30 12:12 new_script
```

Chúng ta có thể thấy rằng các quyền đối với tệp này được đặt thành `664` theo mặc định. Chúng ta chưa cài đặt để tệp này có *quyền thực thi*.

```
$ chmod +x new_script
$ ls -l new_script
-rwxrwxr-x 1 user user 20 Apr 30 12:12 new_script
```

Lệnh này đã cấp quyền thực thi cho *tất cả* mọi người dùng. Hãy lưu ý rằng đây có thể là một rủi ro bảo mật, nhưng hiện tại thì đây là mức quyền có thể chấp nhận được.

```
$ ./new_script
Hello World!
```

Bây giờ chúng ta đã có thể thực thi tệp lệnh của mình.

Định nghĩa Trình thông dịch

Như đã biết, chúng ta có thể chỉ cần nhập văn bản vào một tệp, đặt nó làm tệp thực thi và chạy nó. `new_script` về mặt chức năng vẫn là một tệp văn bản bình thường, nhưng chúng ta đã có thể để cho Bash diễn giải nó. Nhưng nếu tệp lệnh được viết bằng Perl hoặc Python thì sao?

Việc xác định loại trình thông dịch mà ta muốn sử dụng trong dòng đầu tiên của tệp lệnh là một thao tác hữu ích. Dòng này được gọi là *bang line*, hoặc phổ biến hơn là *shebang*. Nó cho hệ thống biết chúng ta muốn tệp này được thực thi như thế nào. Vì đang tìm hiểu về Bash nên chúng ta sẽ sử dụng đường dẫn tuyệt đối đến tệp thực thi Bash, một lần nữa sử dụng ``which``:

```
$ which bash
/bin/bash
```

Shebang bắt đầu bằng dấu thăng và dấu chấm than, theo sau là đường dẫn tuyệt đối ở trên. Hãy mở `new_script` trong một trình soạn thảo văn bản và chèn Shebang. Chúng ta cũng hãy tận dụng cơ hội để chèn một *chú thích* vào tệp lệnh của mình. Chú thích sẽ được trình thông dịch bỏ qua. Chúng được chèn vào vì lợi ích của những người dùng khác muốn hiểu rõ hơn về tệp lệnh của bạn.

```
#!/bin/bash

# This is our first comment. It is also good practice to document all scripts.

echo "Hello World!"
```

Chúng ta cũng sẽ thực hiện một thay đổi bổ sung đối với tên tệp: chúng ta sẽ lưu tệp này dưới dạng `new_script.sh`. Hậu tố tệp ".sh" sẽ không thay đổi việc thực thi tệp theo bất kỳ cách nào. Theo quy ước, các tệp lệnh bash sẽ được gắn nhãn `.sh` hoặc `.bash` để ta có thể nhận dạng chúng dễ dàng hơn, giống như cách mà các tệp lệnh Python thường được xác định bằng hậu tố `.py`.

Trình soạn thảo Văn bản phổ biến

Người dùng Linux thường phải làm việc trong môi trường không có trình soạn thảo văn bản đồ họa. Do vậy, chúng ta nên trở nên quen thuộc với thao tác chỉnh sửa tệp văn bản từ dòng lệnh ở một mức độ nhất định. Hai trong số các trình soạn thảo văn bản phổ biến nhất là `vi` và `nano`.

vi

`vi` là một trình soạn thảo văn bản đáng tin cậy và được cài đặt theo mặc định trên hầu hết mọi hệ

thống Linux hiện có. vi đã cho ra một bản sao có tên vi *IMproved* hoặc vim để bổ sung một số chức năng trong khi vẫn duy trì giao diện vi nguyên bản. Mặc dù việc làm việc với vi gây khó khăn cho người dùng mới nhưng trình chỉnh sửa này rất phổ biến và được yêu thích bởi những người dùng đã cất công tìm hiểu về nhiều các tính năng của nó.

Sự khác biệt quan trọng nhất giữa vi và các ứng dụng như Notepad là vi có ba chế độ khác nhau. Khi khởi động, các phím H, J, K và L được sử dụng để điều hướng chứ không phải để nhập. Trong chế độ điều hướng, bạn có thể nhấn phím I để vào chế độ nhập. Bây giờ thì bạn có thể gõ bình thường. Để thoát khỏi chế độ nhập, hãy nhấn phím Esc để quay lại chế độ điều hướng. Từ chế độ điều hướng, bạn có thể nhấn phím : để vào chế độ lệnh. Từ chế độ này, bạn có thể lưu, xóa, thoát hoặc thay đổi tùy chọn.

Mặc dù vi cần có thời gian để có thể linh hoạt, các chế độ khác nhau có thể kịp thời cho phép một người dùng nhanh nhẹn làm việc hiệu quả hơn so với khi sử dụng các trình chỉnh sửa khác.

nano

nano là một công cụ mới hơn, được thiết kế đơn giản và dễ sử dụng hơn vi. nano không có các chế độ khác nhau. Thay vào đó, khi khởi động, người dùng có thể bắt đầu nhập và sử dụng phím Ctrl để truy cập các công cụ được in ở cuối màn hình.

```
[ Welcome to nano. For basic help, type Ctrl+G. ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line M-E Redo
```

Trình soạn thảo văn bản là vấn đề sở thích cá nhân và trình soạn thảo mà bạn chọn sử dụng sẽ không ảnh hưởng gì đến bài học này. Tuy vậy, việc trở nên quen thuộc và thoải mái với một hoặc nhiều trình soạn thảo văn bản sẽ mang lại cho chúng ta một kết quả xứng đáng trong tương lai.

Biến

Biến là một phần quan trọng của bất kỳ một ngôn ngữ lập trình nào, và Bash cũng không phải là một ngoại lệ. Khi bạn bắt đầu một phiên mới từ cửa sổ dòng lệnh, vỏ đã đặt sẵn một số biến cho bạn. Biến PATH là một ví dụ về điều này. Ta gọi đây là các biến môi trường vì chúng thường xác định các đặc điểm của môi trường vỏ của chúng ta. Các biến môi trường có thể được thêm vào và sửa đổi, nhưng bây giờ ta nên tập trung vào việc thiết lập các biến bên trong tệp lệnh.

Ta sẽ sửa đổi tệp lệnh của mình để nó trông như thế này:

```
#!/bin/bash
```

```
# This is our first comment. It is also good practice to document all scripts.

username=Carol

echo "Hello $username!"
```

Trong trường hợp này, chúng ta đã tạo ra một *biến* được gọi là `username` và ta đã gán cho nó *giá trị* của `Carol`. Hãy lưu ý rằng không có khoảng cách giữa tên biến, dấu bằng hoặc giá trị được gán.

Trong dòng tiếp theo, chúng ta đã sử dụng lệnh `echo` với biến, nhưng có một ký hiệu đô la (\$) trước tên biến. Điều này rất quan trọng vì nó cho vở biết rằng chúng ta muốn coi `username` là một biến chứ không chỉ là một từ bình thường. Bằng cách nhập `$username` vào lệnh, vở sẽ hiểu là ta muốn thực hiện *phép thay thế*, tức thay thế *tên* của một biến bằng *giá trị* được gán cho biến đó.

Khi thực thi tệp lệnh mới, chúng ta sẽ nhận được kết quả này:

```
$ ./new_script.sh
Hello Carol!
```

- Tên biến chỉ được chứa ký tự chữ và số hoặc dấu gạch dưới và phải phân biệt chữ hoa chữ thường. `Username` và `username` sẽ được coi là hai biến khác nhau.
- Biến thay thế cũng có thể có định dạng `${username}` bằng cách thêm `{ }`. Điều này cũng được chấp nhận.
- Các biến trong Bash có *kiểu ngầm định* và được coi là các chuỗi. Điều này có nghĩa là việc thực hiện các hàm toán học trong Bash sẽ phức tạp hơn so với trong các ngôn ngữ lập trình khác như C/C++:

```
#!/bin/bash

# This is our first comment. It is also good practice to document all scripts.

username=Carol
x=2
y=4
z=$((x+y))
echo "Hello $username!"
echo "$x + $y"
echo "$z"
```

```
$ ./new_script.sh
Hello Carol!
2 + 4
2+4
```

Sử dụng Trích dẫn với các Biến

Hãy thực hiện thay đổi sau đối với giá trị của biến `username`:

```
#!/bin/bash

# This is our first comment. It is also good practice to document all scripts.

username=Carol Smith

echo "Hello $username!"
```

Việc chạy tệp lệnh này sẽ cho ra lỗi:

```
$ ./new_script.sh
./new_script.sh: line 5: Smith: command not found
Hello !
```

Hãy nhớ rằng Bash là một trình thông dịch và do đó, nó *thông dịch* từng dòng của tệp lệnh. Trong trường hợp này, nó diễn giải chính xác `username=Carol` là đang đặt một biến `username` với giá trị `Carol`. Nhưng sau đó, nó sẽ diễn giải khoảng trắng là biểu thị kết thúc nhiệm vụ đó, và `Smith` là tên của một lệnh. Để có khoảng trắng và tên `Smith` được đưa vào làm giá trị mới của biến, chúng ta sẽ phải đặt dấu trích dẫn kép (") bên ngoài `Carol Smith`.

```
#!/bin/bash

# This is our first comment. It is also good practice to document all scripts.

username="Carol Smith"

echo "Hello $username!"
```

```
$ ./new_script.sh
```



```
Hello Carol Smith!
```

Một điều quan trọng cần lưu ý trong Bash là dấu trích dẫn kép và trích dẫn đơn (') hoạt động theo hai cách hoàn toàn khác nhau. Dấu trích dẫn kép được coi là “yếu” vì chúng cho phép trình thông dịch thực hiện thay thế bên trong dấu. Dấu trích dẫn đơn được coi là “mạnh” vì chúng ngăn không cho bất kỳ sự thay thế nào xảy ra. Hãy xem ví dụ sau:

```
#!/bin/bash

# This is our first comment. It is also good practice to document all scripts.

username="Carol Smith"

echo "Hello $username!"
echo 'Hello $username!'
```

```
$ ./new_script.sh
Hello Carol Smith!
Hello $username!
```

Trong lệnh `echo` thứ hai, trình thông dịch đã bị ngăn không cho thay thế `$username` bằng `Carol Smith`, vì vậy nên đầu ra sẽ được hiểu theo nghĩa đen.

Đối số

Bạn đã quen với việc sử dụng các đối số trong các tiện ích cốt lõi của Linux. Ví dụ: `rm testfile` chứa cả tệp thực thi `rm` và một đối số `testfile`. Các đối số có thể được chuyển đến tệp lệnh khi thực thi và sẽ sửa đổi cách tệp lệnh hoạt động. Chúng có thể được thực hiện một cách dễ dàng.

```
#!/bin/bash

# This is our first comment. It is also good practice to document all scripts.

username=$1

echo "Hello $username!"
```

Thay vì gán một giá trị cho `username` ngay bên trong tệp lệnh, chúng ta đang gán cho nó giá trị của một biến mới là `$1`, tức giá trị của *đối số đầu tiên*.

```
$ ./new_script.sh Carol
Hello Carol!
```

Chín đối số đầu tiên sẽ được xử lý theo cách này. Có nhiều cách để xử lý hơn chín đối số, nhưng điều đó nằm ngoài phạm vi của bài học này. Chúng ta sẽ minh họa một ví dụ chỉ sử dụng hai đối số:

```
#!/bin/bash

# This is our first comment. It is also good practice to document all scripts.

username1=$1
username2=$2
echo "Hello $username1 and $username2!"
```

```
$ ./new_script.sh Carol Dave
Hello Carol and Dave!
```

Có một việc quan trọng cần phải cân nhắc khi sử dụng đối số: Trong ví dụ trên, có hai đối số là `Carol` và `Dave` tương ứng được gán cho `$1` và `$2`. Ví dụ, nếu đối số thứ hai bị thiếu, vỏ sẽ không đưa ra lỗi. Giá trị của `$2` sẽ đơn giản là *null* hoặc không có gì cả.

```
$ ./new_script.sh Carol
Hello Carol and !
```

Trong trường hợp của chúng ta, việc đưa một số logic vào tệp lệnh để các *điều kiện* khác nhau ảnh hưởng đến *đầu ra* mà ta muốn in là một ý hay. Chúng ta sẽ bắt đầu bằng cách giới thiệu một biến hữu ích khác và sau đó chuyển sang tạo các câu lệnh *if*.

Trả về Số của Đối số

Trong khi các biến như `$1` và `$2` chứa giá trị của các đối số vị trí thì một biến khác là `$#` sẽ chứa *số lượng đối số*.

```
#!/bin/bash

# This is our first comment. It is also good practice to document all scripts.
```

```
username=$1
```

```
echo "Hello $username!"
```

```
echo "Number of arguments: $#."
```

```
$ ./new_script.sh Carol Dave
```

```
Hello Carol!
```

```
Number of arguments: 2.
```

Logic có điều kiện

Việc sử dụng logic có điều kiện trong lập trình là một chủ đề rộng và sẽ không được đề cập sâu trong bài học này. Chúng ta sẽ tập trung vào *cú pháp* của các điều kiện trong Bash - yếu tố khác biệt so với hầu hết các ngôn ngữ lập trình khác.

Hãy bắt đầu bằng cách xem xét những gì chúng ta muốn đạt được. Chúng ta có một tệp lệnh đơn giản có thể in lời chào cho một người dùng. Nếu có bất kỳ ai khác ngoài một người dùng, chúng ta sẽ in một thông báo lỗi.

- *Điều kiện* mà chúng ta đang thử là số lượng người dùng được chứa trong biến `$#`. Chúng ta muốn biết liệu giá trị của `$#` có phải là 1 hay không.
- Nếu điều kiện là *đúng* thì *hành động* chúng ta sẽ thực hiện là chào người dùng.
- Nếu điều kiện là *sai*, chúng ta sẽ in thông báo lỗi. Bây giờ logic đã rõ ràng, chúng ta sẽ tập trung vào *cú pháp* cần thiết để triển khai logic này.

```
#!/bin/bash

# A simple script to greet a single user.

if [ $# -eq 1 ]
then
    username=$1

    echo "Hello $username!"
else
    echo "Please enter only one argument."
fi
echo "Number of arguments: $#."
```

Logic điều kiện được chứa ở giữa `if` và `fi`. Điều kiện để kiểm tra nằm giữa các dấu ngoặc vuông [] và hành động cần thực hiện nếu điều kiện là đúng được biểu thị sau `then`. Lưu ý khoảng cách giữa các dấu ngoặc vuông và logic chứa trong đó. Việc bỏ qua khoảng cách này sẽ gây ra lỗi.

Tập lệnh này sẽ xuất ra lời chào của chúng ta *hoặc* thông báo lỗi. Nhưng nó sẽ luôn in dòng `Number of arguments` (Số của Đối số).

```
$ ./new_script.sh
Please enter only one argument.
Number of arguments: 0.
$ ./new_script.sh Carol
Hello Carol!
Number of arguments: 1.
```

Lưu ý câu lệnh `if`. Chúng ta đã sử dụng `-eq` để thực hiện *so sánh số*. Trong trường hợp này, ta đang kiểm tra xem giá trị của `$#` có *bằng* với 1 hay không. Các so sánh khác chúng ta có thể thực hiện là:

-ne

Không bằng

-gt

Lớn hơn

-ge

Lớn hơn hoặc bằng

-lt

Ít hơn

-le

Ít hơn hoặc bằng

Bài tập Hướng dẫn

1. Người dùng nhập nội dung sau vào vỏ của họ:

```
$ PATH=~/.scripts
$ ls
Command 'ls' is available in '/bin/ls'
The command could not be located because '/bin' is not included in the PATH environment
variable.
ls: command not found
```

- Người dùng đã làm gì?

- Lệnh nào sẽ kết hợp giá trị hiện tại của PATH với thư mục mới ~/.scripts?

2. Hãy xem xét các tệp lệnh sau đây. Hãy lưu ý rằng nó đang sử dụng `elif` để kiểm tra điều kiện thứ hai:

```
> /bin/bash

> fruit1 = Apples
> fruit2 = Oranges

if [ $1 -lt $# ]
then
    echo "This is like comparing $fruit1 and $fruit2!"
> elif [ $1 -gt $2 ]
then
>   echo '$fruit1 win!'
else
>   echo "Fruit2 win!"
> done
```

- Các dòng được đánh dấu > chứa lỗi. Hãy tiến hành sửa lỗi.

3. Đầu ra sẽ là gì trong các tình huống sau đây?

```
$ ./guided1.sh 3 0
```

```
$ ./guided1.sh 2 4
```

```
$ ./guided1.sh 0 1
```

Bài tập Mở rộng

1. Hãy viết một tệp lệnh đơn giản để kiểm tra xem có chính xác là hai đối số đã được thông qua hay không. Nếu đúng, hãy in các đối số theo thứ tự ngược lại. Hãy xem ví dụ này (lưu ý: mã của bạn có thể trông khác với mã này nhưng vẫn nên dẫn đến cùng một đầu ra):

```
if [ $1 == $number ]
then
  echo "True!"
fi
```

2. Đoạn mã này là chính xác nhưng nó không phải là một phép so sánh số. Hãy thử tìm kiếm trên internet để khám phá xem mã này khác với việc sử dụng `-eq` như thế nào.

3. Có một biến môi trường sẽ in thư mục hiện tại. Hãy sử dụng `env` để khám phá tên của biến này.

4. Sử dụng những gì bạn đã học được trong câu hỏi 2 và 3, hãy viết một tệp lệnh ngăn chấp nhận một đối số. Nếu một đối số được thông qua, hãy kiểm tra xem đối số đó có khớp với tên của thư mục hiện tại không. Nếu đúng, hãy in `yes`. Nếu sai, hãy in `no`.

Tóm tắt

Trong bài học này, bạn đã học được:

- Cách tạo và thực thi các tập lệnh đơn giản
- Cách sử dụng shebang để chỉ định trình thông dịch
- Cách đặt và sử dụng các biến bên trong tập lệnh
- Cách xử lý đối số trong tập lệnh
- Cách xây dựng câu lệnh `if`
- Cách so sánh các số bằng toán tử số Các lệnh được dùng trong bài tập:

`echo`

In một chuỗi ra đầu ra tiêu chuẩn.

`env`

In tất cả các biến môi trường thành đầu ra tiêu chuẩn.

`which`

In đường dẫn tuyệt đối của một lệnh.

`chmod`

Thay đổi quyền của một tệp.

Các biến đặc biệt dùng trong bài tập:

`$1, $2, ... $9`

Chứa các đối số vị trí được chuyển đến tập lệnh.

`$#`

Chứa số lượng đối số được truyền cho tập lệnh.

`$PATH`

Chứa các thư mục có các tệp thực thi được sử dụng bởi hệ thống.

Các toán tử dùng trong bài tập: `-ne::` Không bằng `-gt::` Lớn hơn `-ge::` Lớn hơn hoặc bằng `-lt::` Ít hơn `-le::` Ít hơn hoặc bằng

Đáp án Bài tập Hướng dẫn

1. Người dùng nhập nội dung sau vào vỏ của họ:

```
$ PATH=~/.scripts
$ ls
Command 'ls' is available in '/bin/ls'
The command could not be located because '/bin' is not included in the PATH environment
variable.
ls: command not found
```

- Người dùng đã làm gì?

Người dùng đã ghi đè nội dung của PATH bằng thư mục `~/scripts`. Sẽ không thể tìm thấy lệnh `ls` nữa vì lệnh này không có trong PATH. Hãy lưu ý rằng thay đổi này chỉ ảnh hưởng đến phiên làm việc hiện tại, nếu bạn đăng xuất và đăng nhập lại sẽ hoàn nguyên thay đổi.

- Lệnh nào sẽ kết hợp giá trị hiện tại của PATH với thư mục mới `~/scripts`?

```
PATH=$PATH:~/scripts
```

2. Hãy xem xét các tệp lệnh sau đây. Hãy lưu ý rằng nó đang sử dụng `elif` để kiểm tra điều kiện thứ hai:

```
> /!bin/bash

> fruit1 = Apples
> fruit2 = Oranges

if [ $1 -lt $# ]
then
    echo "This is like comparing $fruit1 and $fruit2!"
> elif [$1 -gt $2 ]
then
> echo '$fruit1 win!'
else
> echo "Fruit2 win!"
> done
```

- Các dòng được đánh dấu `>` chứa lỗi. Hãy tiến hành sửa lỗi.

```
#!/bin/bash

fruit1=Apples
fruit2=Oranges

if [ $1 -lt $# ]
then
    echo "This is like comparing $fruit1 and $fruit2!"
elif [ $1 -gt $2 ]
then
    echo "$fruit1 win!"
else
    echo "$fruit2 win!"
fi
```

3. Đầu ra sẽ là gì trong các tình huống sau đây?

```
$ ./guided1.sh 3 0
```

Apples win!

```
$ ./guided1.sh 2 4
```

Oranges win!

```
$ ./guided1.sh 0 1
```

This is like comparing Apples and Oranges!

Đáp án Bài tập Mở rộng

- Hãy viết một tệp lệnh đơn giản để kiểm tra xem có chính xác là hai đối số đã được thông qua hay không. Nếu đúng, hãy in các đối số theo thứ tự ngược lại. Hãy xem ví dụ này (lưu ý: mã của bạn có thể trông khác với mã này nhưng vẫn nên dẫn đến cùng một đầu ra):

```
if [ $1 == $number ]
then
  echo "True!"
fi
```

```
#!/bin/bash

if [ $# -ne 2 ]
then
  echo "Error"
else
  echo "$2 $1"
fi
```

- Đoạn mã này là chính xác nhưng nó không phải là một phép so sánh số. Hãy thử tìm kiếm trên internet để khám phá xem mã này khác với việc sử dụng `-eq` như thế nào.

Sử dụng `==` sẽ so sánh *các chuỗi*. Nghĩa là nếu các ký tự của cả hai biến khớp chính xác thì điều kiện là đúng.

<code>abc == abc</code>	<i>true</i>
<code>abc == ABC</code>	<i>false</i>
<code>1 == 1</code>	<i>true</i>
<code>1+1 == 2</code>	<i>false</i>

So sánh chuỗi dẫn đến hành vi không mong muốn nếu bạn đang kiểm tra số.

- Có một biến môi trường sẽ in thư mục hiện tại. Hãy sử dụng `env` để khám phá tên của biến này.

`PWD`

- Sử dụng những gì bạn đã học được trong câu hỏi 2 và 3, hãy viết một tệp lệnh ngăn chấp nhận một đối số. Nếu một đối số được thông qua, hãy kiểm tra xem đối số đó có khớp với tên của thư

mục hiện tại không. Nếu đúng, hãy in `yes`. Nếu sai, hãy in `no`.

```
#!/bin/bash

if [ "$1" == "$PWD" ]
then
    echo "yes"
else
    echo "no"
fi
```



3.3 Bài 2

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	3 Sức mạnh của Dòng lệnh
Mục tiêu:	3.3 Biến các Lệnh thành một Tệp lệnh
Bài:	2 trên 2

Giới thiệu

Trong phần trước, chúng ta đã sử dụng ví dụ đơn giản này để minh họa tệp lệnh Bash:

```
#!/bin/bash

# A simple script to greet a single user.

if [ $# -eq 1 ]
then
    username=$1

    echo "Hello $username!"
else
    echo "Please enter only one argument."
fi
echo "Number of arguments: $#."
```

- Tất cả các tệp lệnh phải bắt đầu bằng *shebang* để xác định đường dẫn đến trình thông dịch.

- Tất cả các tập lệnh nên bao gồm các chú thích để mô tả việc sử dụng chúng.
- Tập lệnh cụ thể này làm việc với *đối số*; đối số sẽ được chuyển đến tập lệnh khi nó được gọi.
- Tập lệnh này chứa câu lệnh *if*; câu lệnh này kiểm tra các điều kiện của biến tích hợp `$#`. Biến này sẽ được đặt thành số lượng của đối số.
- Nếu số lượng của đối số được truyền cho tập lệnh bằng 1 thì giá trị của đối số đầu tiên sẽ được truyền cho một biến mới có tên `username` và tập lệnh sẽ lặp lại lời chào tới người dùng. Nếu ngược lại, một thông báo lỗi sẽ được hiển thị.
- Cuối cùng, tập lệnh sẽ lặp lại số lượng đối số. Điều này rất hữu ích trong việc gỡ lỗi.

Đây là một ví dụ hữu ích để bắt đầu giải thích một số tính năng khác của tập lệnh Bash.

Mã Thoát

Bạn sẽ nhận thấy rằng tập lệnh của chúng ta có hai trạng thái có thể xảy ra: một là nó sẽ in ra "Hello <user>!", hai là sẽ in ra một thông báo lỗi. Điều này khá là bình thường đối với nhiều tiện ích cốt lõi của chúng ta. Hãy thử `cat` - lệnh mà chắc chắn bạn đang dần trở nên quen thuộc.

Hãy so sánh việc sử dụng thành công `cat` với tình huống sử dụng thất bại. Hãy lưu ý rằng ví dụ của chúng ta ở trên là một tập lệnh có tên `new_script.sh`.

```
$ cat -n new_script.sh

1  #!/bin/bash
2
3  # A simple script to greet a single user.
4
5  if [ $# -eq 1 ]
6  then
7      username=$1
8
9      echo "Hello $username!"
10 else
11     echo "Please enter only one argument."
12 fi
13 echo "Number of arguments: $#."
```

Lệnh này thành công và bạn sẽ nhận thấy rằng cờ `-n` cũng đã in số dòng. Chúng rất hữu ích trong việc gỡ lỗi tập lệnh, nhưng hãy lưu ý rằng chúng *không* phải là một phần của tập lệnh.

Bây giờ, chúng ta sẽ kiểm tra giá trị của một biến tích hợp sẵn mới là `$?`. Hiện tại, ta hãy chỉ chú ý

đến đầu ra:

```
$ echo $?  
0
```

Bây giờ, hãy xem xét tình huống mà trong đó `cat` bị lỗi. Đầu tiên, chúng ta sẽ thấy một thông báo lỗi; sau đó, ta sẽ kiểm tra giá trị của `$?`.

```
$ cat -n dummyfile.sh  
cat: dummyfile.sh: No such file or directory  
$ echo $?  
1
```

Lời giải thích cho hành vi này là bởi bất kỳ tác vụ thực thi nào của tiện ích `cat` cũng sẽ trả về một *mã thoát*. Mã thoát sẽ cho chúng ta biết lệnh đã thành công hay đã gặp lỗi. Mã thoát `0` cho ta biết lệnh đã hoàn tất thành công. Điều này đúng với hầu hết mọi lệnh Linux mà bạn sẽ làm việc cùng. Bất kỳ mã thoát nào khác đều sẽ chỉ ra một loại lỗi nào đó. Mã thoát của *lệnh chạy cuối cùng* sẽ được lưu trữ trong biến `$?`.

Người dùng thường sẽ không thấy được mã thoát, nhưng chúng lại rất hữu ích khi viết tệp lệnh. Giả sử một tệp lệnh mà trong đó chúng ta có thể sao chép tệp vào một ổ đĩa mạng từ xa. Có nhiều nguyên nhân khiến tác vụ sao chép có thể không thành công, ví dụ như khi máy cục bộ của chúng ta không được kết nối với mạng hoặc ổ đĩa từ xa có thể đã đầy. Bằng cách kiểm tra mã thoát của tiện ích sao chép, chúng ta có thể cảnh báo người dùng về các sự cố khi chạy tệp lệnh.

Triển khai mã thoát là một bài tập thực hành rất hữu ích. Vì vậy, chúng ta sẽ thực hiện nó ngay bây giờ. Ta có hai hướng trong tệp lệnh, thành công và thất bại. Hãy sử dụng số `0` (không) để biểu thị thành công và `1` (một) để biểu thị thất bại.

```
1 #!/bin/bash  
2  
3 # A simple script to greet a single user.  
4  
5 if [ $# -eq 1 ]  
6 then  
7     username=$1  
8  
9     echo "Hello $username!"  
10    exit 0  
11 else  
12    echo "Please enter only one argument."
```

```

13     exit 1
14 fi
15 echo "Number of arguments: $#."

```

```

$ ./new_script.sh Carol
Hello Carol!
$ echo $?
0

```

Hãy lưu ý rằng lệnh `echo` trên dòng 15 đã bị bỏ qua hoàn toàn. Việc sử dụng `exit` sẽ kết thúc tệp lệnh ngay lập tức và giúp ta không gặp lại dòng đó nữa.

Xử lý nhiều Đối số

Cho đến nay, tệp lệnh của chúng ta chỉ có thể xử lý từng tên người dùng một. Bất kỳ số lượng đối số nào ngoài một sẽ gây ra lỗi. Hãy cùng khám phá những cách chúng ta có thể dùng để làm cho tệp lệnh này linh hoạt hơn.

Phản ứng bản năng đầu tiên của người dùng có thể sẽ là sử dụng nhiều biến vị trí hơn như `$2`, `$3`, v.v. Thật không may là chúng ta không thể đoán trước số lượng đối số mà người dùng có thể chọn để sử dụng. Một cách hữu ích trong việc giải quyết vấn đề này là tìm hiểu về nhiều biến tích hợp hơn.

Chúng ta sẽ thay đổi logic của tệp lệnh: không có đối số nào sẽ gây ra lỗi, nhưng với bất kỳ số lượng đối số nào khác thì lệnh sẽ thành công. Tệp lệnh mới này sẽ được gọi là `friendly2.sh`.

```

1  #!/bin/bash
2
3  # a friendly script to greet users
4
5  if [ $# -eq 0 ]
6  then
7      echo "Please enter at least one user to greet."
8      exit 1
9  else
10     echo "Hello $@"
11     exit 0
12 fi

```

```

$ ./friendly2.sh Carol Dave Henry

```



```
Hello Carol Dave Henry!
```

Có hai biến tích hợp có chứa tất cả các đối số được chuyển đến tệp lệnh: `$@` và `$*`. Cả hai tệp lệnh này đều sẽ hoạt động như nhau trong hầu hết các trường hợp. Bash sẽ *phân tích cú pháp* của các đối số và tách từng đối số khi nó gặp một khoảng cách giữa chúng. Trên thực tế, nội dung của `$@` sẽ trông như thế này:

0	1	2
Carol	Dave	Henry

Nếu bạn đã quen thuộc với các ngôn ngữ lập trình khác, bạn có thể nhận ra đây là biến *mảng* (array). Mảng trong Bash có thể được tạo đơn giản bằng cách đặt khoảng trắng giữa các phần tử như biến `FILES` trong tệp lệnh `arraytest` bên dưới:

```
FILES="/usr/sbin/accept /usr/sbin/pwck/ usr/sbin/chroot"
```

Nó chứa một danh sách với nhiều mục. Cho đến nay, điều này không được coi là hữu ích lắm vì chúng ta vẫn chưa có cách nào để xử lý các mục này một cách riêng lẻ.

Vòng lặp For

Hãy tham khảo ví dụ `arraytest` ở trên. Nếu bạn còn nhớ, trong ví dụ này, chúng ta đang chỉ định một mảng của riêng chúng ta có tên là `FILES`. Điều ta cần là một cách để “giải nén” biến này và lần lượt truy cập từng giá trị riêng lẻ. Để làm được điều đó, chúng ta phải sử dụng một cấu trúc gọi là *vòng lặp for* (for loop). Cấu trúc này có trong tất cả các ngôn ngữ lập trình. Có hai biến mà chúng ta sẽ đề cập đến: một là phạm vi, hai giá trị riêng lẻ mà chúng ta đang thực hiện. Sau đây là toàn bộ tệp lệnh:

```
#!/bin/bash

FILES="/usr/sbin/accept /usr/sbin/pwck/ usr/sbin/chroot"

for file in $FILES
do
  ls -lh $file
done
```

```
$ ./arraytest
lrwxrwxrwx 1 root root 10 Apr 24 11:02 /usr/sbin/accept -> cupsaccept
```

```
-rwxr-xr-x 1 root root 54K Mar 22 14:32 /usr/sbin/pwck
-rwxr-xr-x 1 root root 43K Jan 14 07:17 /usr/sbin/chroot
```

Nếu bạn tham khảo lại ví dụ `friendly2.sh` ở trên, bạn có thể thấy rằng chúng ta đang làm việc với một loạt các giá trị chứa trong một biến duy nhất là `$@`. Để rõ ràng hơn, ta sẽ gọi biến thứ hai là `username`. Tập lệnh của chúng ta bây giờ sẽ trông như thế này:

```
1 #!/bin/bash
2
3 # a friendly script to greet users
4
5 if [ $# -eq 0 ]
6 then
7     echo "Please enter at least one user to greet."
8     exit 1
9 else
10    for username in $@
11    do
12        echo "Hello $username!"
13    done
14    exit 0
15 fi
```

Hãy nhớ rằng biến mà bạn xác định ở đây có thể được đặt tên theo bất kỳ tên nào mà bạn muốn, và rằng tất cả các dòng bên trong `do... done` sẽ được thực thi một lần cho mỗi phần tử của mảng. Hãy quan sát đầu ra từ tập lệnh của chúng ta:

```
$ ./friendly2.sh Carol Dave Henry
Hello Carol!
Hello Dave!
Hello Henry!
```

Bây giờ, hãy giả sử rằng chúng ta muốn làm cho đầu ra của mình trông có vẻ giống lời chào của con người hơn một chút, cụ thể là lời chào chỉ nằm trên một dòng.

```
1 #!/bin/bash
2
3 # a friendly script to greet users
4
5 if [ $# -eq 0 ]
6 then
```

```

7   echo "Please enter at least one user to greet."
8   exit 1
9   else
10  echo -n "Hello $1"
11  shift
12  for username in $@
13  do
14      echo -n ", and $username"
15  done
16  echo "!"
17  exit 0
18  fi

```

Một vài lưu ý:

- Sử dụng `-n` với `echo` sẽ *loại bỏ ký tự xuống dòng* sau khi in. Điều này có nghĩa là tất cả các `echo` sẽ in trên cùng một dòng, và dòng mới sẽ chỉ được in sau dấu `!` trên dòng 16.
- Lệnh `shift` sẽ xóa phần tử đầu tiên của mảng, để:

0	1	2
Carol	Dave	Henry

trở thành:

0	1
Dave	Henry

Hãy quan sát đầu ra:

```

$ ./friendly2.sh Carol
Hello Carol!
$ ./friendly2.sh Carol Dave Henry
Hello Carol, and Dave, and Henry!

```

Sử dụng Biểu thức Chính quy để Thực hiện Kiểm tra Lỗi

Có thể chúng ta muốn xác minh tất cả các đối số mà người dùng đang nhập. Ví dụ: có lẽ ta muốn đảm bảo rằng tất cả các tên được chuyển đến `friendly2.sh` sẽ chỉ chứa *các chữ cái*, và mọi ký tự hoặc số đặc biệt đều sẽ gây ra lỗi. Để thực hiện kiểm tra lỗi này, ta sẽ sử dụng `grep`.

Hãy nhớ lại rằng chúng ta có thể sử dụng biểu thức chính quy với `grep`.

```
$ echo Animal | grep "^[A-Za-z]*$"
Animal
$ echo $?
0
```

```
$ echo 4n1m1 | grep "^[A-Za-z]*$"
$ echo $?
1
```

`^` và `$` lần lượt chỉ ra phần đầu và phần cuối của dòng. `[A-Za-z]` cho biết phạm vi chữ cái, chữ hoa hoặc chữ thường. `*` là *định lượng* và nó sẽ sửa đổi phạm vi chữ cái của chúng ta sao cho số 0 khớp với nhiều chữ cái. Tóm lại, `grep` của chúng ta sẽ thành công nếu đầu vào *chỉ* có chữ cái và sẽ không thành công trong trường hợp ngược lại.

Điều tiếp theo cần lưu ý là `grep` đang trả lại mã thoát dựa trên việc có khớp hay không. Kết quả khớp sẽ trả về 0 và kết quả không khớp sẽ trả về 1. Chúng ta có thể sử dụng điều này để kiểm tra các đối số bên trong tệp lệnh của mình.

```
1 #!/bin/bash
2
3 # a friendly script to greet users
4
5 if [ $# -eq 0 ]
6 then
7     echo "Please enter at least one user to greet."
8     exit 1
9 else
10    for username in $@
11    do
12        echo $username | grep "^[A-Za-z]*$" > /dev/null
13        if [ $? -eq 1 ]
14        then
15            echo "ERROR: Names must only contains letters."
16            exit 2
17        else
18            echo "Hello $username!"
19        fi
20    done
21    exit 0
```

22 fi

Ở dòng 12, chúng ta đang chuyển hướng đầu ra tiêu chuẩn sang `/dev/null`, đây là một cách đơn giản để chặn nó. Chúng ta không muốn thấy bất kỳ đầu ra nào từ lệnh `grep` mà chỉ muốn kiểm tra mã thoát của nó xảy ra trên dòng 13. Cũng lưu ý rằng chúng ta đang sử dụng mã thoát 2 để biểu thị một đối số không hợp lệ. Thông thường, ta nên sử dụng các mã thoát khác nhau để chỉ ra các lỗi khác nhau; theo cách này, người dùng có kinh nghiệm sẽ có thể sử dụng các mã thoát này để khắc phục sự cố.

```
$ ./friendly2.sh Carol Dave Henry
Hello Carol!
Hello Dave!
Hello Henry!
$ ./friendly2.sh 42 Carol Dave Henry
ERROR: Names must only contains letters.
$ echo $?
2
```

Bài tập Hướng dẫn

1. Hãy đọc nội dung của `script1.sh` bên dưới:

```
#!/bin/bash

if [ $# -lt 1 ]
then
    echo "This script requires at least 1 argument."
    exit 1
fi

echo $1 | grep "[A-Z]*$" > /dev/null
if [ $? -ne 0 ]
then
    echo "no cake for you!"
    exit 2
fi

echo "here's your cake!"
exit 0
```

Đầu ra của các lệnh này là gì?

- `./script1.sh`

- `echo $?`

- `./script1.sh cake`

- `echo $?`

- `./script1.sh CAKE`

- `echo $?`

2. Hãy đọc nội dung của tệp `script2.sh`:

```
for filename in $1/*.txt
do
    cp $filename $filename.bak
done
```

Hãy mô tả mục đích của tệp lệnh này theo ý hiểu của bạn.

Bài tập Mở rộng

1. Hãy tạo một tệp lệnh sẽ lấy bất kỳ số lượng đối số nào từ người dùng và chỉ in những đối số đó là các số lớn hơn 10.

Tóm tắt

Trong bài học này, bạn đã học về:

- Mã thoát là gì, ý nghĩa và cách triển khai chúng.
- Cách kiểm tra mã thoát của lệnh.
- Vòng lặp `for` là gì và cách sử dụng chúng với mảng.
- Cách sử dụng `grep`, biểu thức chính quy và mã thoát để kiểm tra đầu vào của người dùng trong tệp lệnh.

Các lệnh được dùng trong bài tập:

`shift`

Loại bỏ phần tử đầu tiên của một mảng.

Các biến đặc biệt:

`$?`

Chứa mã thoát của lệnh cuối cùng được thực thi.

`$@`, `$*`

Chứa tất cả các đối số được chuyển đến tệp lệnh dưới dạng một mảng.

Đáp án Bài tập Hướng dẫn

1. Hãy đọc nội dung của `script1.sh` bên dưới:

```
#!/bin/bash

if [ $# -lt 1 ]
then
    echo "This script requires at least 1 argument."
    exit 1
fi

echo $1 | grep "^[A-Z]*$" > /dev/null
if [ $? -ne 0 ]
then
    echo "no cake for you!"
    exit 2
fi

echo "here's your cake!"
exit 0
```

Đầu ra của các lệnh này là gì?

- Lệnh: `./script1.sh`

Đầu ra: `This script requires at least 1 argument.`

- Lệnh: `echo $?`

Đầu ra: `1`

- Lệnh: `./script1.sh cake`

Đầu ra: `no cake for you!`

- Lệnh: `echo $?`

Đầu ra: `2`

- Lệnh: `./script1.sh CAKE`

Đầu ra: `here's your cake!`

- Lệnh: `echo $?`

Đầu ra: `0`

2. Hãy đọc nội dung của tệp `script2.sh`:

```
for filename in $1/*.txt
do
    cp $filename $filename.bak
done
```

Hãy mô tả mục đích của tệp lệnh này theo ý hiểu của bạn.

Tệp lệnh này sẽ tạo các bản sao dự phòng của tất cả các tệp kết thúc bằng `.txt` trong thư mục con được xác định trong đối số đầu tiên.

Đáp án Bài tập Mở rộng

1. Hãy tạo một tệp lệnh sẽ lấy bất kỳ số lượng đối số nào từ người dùng và chỉ in những đối số đó là các số lớn hơn 10.

```
#!/bin/bash

for i in $@
do
  echo $i | grep "^[0-9]*$" > /dev/null
  if [ $? -eq 0 ]
  then
    if [ $i -gt 10 ]
    then
      echo -n "$i "
    fi
  fi
done
echo ""
```



Chủ đề 4: Hệ điều hành Linux



**Linux
Professional
Institute**

4.1 Chọn Hệ điều hành

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 4.1](#)

Khối lượng

1

Các lĩnh vực kiến thức chính

- Sự khác biệt giữa Windows, OS X và Linux
- Quản lý vòng đời của bản phân phối

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

- GUI so với dòng lệnh, cấu hình máy tính để bàn
- Chu kỳ bảo trì, beta và ổn định



4.1 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	4 Hệ điều hành Linux
Mục tiêu:	4.1 Lựa chọn Hệ điều hành
Bài:	1 trên 1

Giới thiệu

Bất kể bạn đang sử dụng hệ thống máy tính của mình ở nhà, ở trường đại học hay trong doanh nghiệp, bạn vẫn sẽ phải đưa ra quyết định xem mình sẽ sử dụng hệ điều hành nào. Bạn có thể tự lựa chọn hệ điều hành, đặc biệt nếu đó là máy tính cá nhân của bạn; nhưng bạn cũng có thể sẽ là người chịu trách nhiệm đưa ra lựa chọn trong hệ thống trong doanh nghiệp của mình. Như đối với tất cả những vấn đề khác, việc hiểu rõ về các lựa chọn sẵn có sẽ giúp bạn đưa ra được một quyết định đáp ứng đủ các nhu cầu trách nhiệm. Trong bài học này, chúng tôi mong muốn giúp bạn cập nhật thông tin về những lựa chọn về hệ điều hành mà có thể bạn đang cân nhắc.

Hệ Điều hành là gì

Để bắt đầu hành trình lựa chọn một hệ điều hành, việc đầu tiên là ta phải hiểu được thuật ngữ này có nghĩa là gì. Hệ điều hành nằm ở trung tâm máy tính của bạn và cho phép các ứng dụng chạy bên trong và bên trên nó. Ngoài ra, hệ điều hành sẽ chứa các trình điều khiển để truy cập vào phần cứng của máy tính như ổ đĩa và các phân vùng đĩa, màn hình, bàn phím, card mạng, v.v. Chúng ta thường viết tắt hệ điều hành thành OS (Operating System). Ngày nay, có rất nhiều hệ điều hành có sẵn cho cả máy tính trong doanh nghiệp cũng như những máy tính tại gia. Để làm cho việc lựa chọn trở nên đơn giản hơn, chúng ta có thể nhóm các sự lựa chọn như sau:

- Hệ điều hành dựa trên Linux
 - Linux Doanh nghiệp
 - Linux Người tiêu dùng
- UNIX
- Hệ điều hành Mac
- Hệ điều hành dựa trên Windows
 - Windows Máy chủ
 - Windows Máy tính để bàn

Chọn một Bản Phân phối Linux

Nhân Linux và các Bản Phân phối Linux

Khi nói về các bản phân phối Linux, hệ điều hành của chúng là Linux. Linux là *hạt nhân* và là cốt lõi của mọi bản phân phối Linux. Phần mềm của nhân Linux được duy trì bởi một nhóm các cá nhân do Linus Torvalds đứng đầu. Torvalds được một tập đoàn công nghiệp có tên là Quỹ Linux (The Linux Foundation) tuyển dụng để phát triển nhân Linux.

NOTE

Nhân Linux được bắt đầu phát triển vào năm 1991 bởi Linus Torvalds - một sinh viên đến từ Phần Lan. Năm 1992, bản phát hành Hạt nhân đầu tiên theo Giấy phép Công cộng GNU phiên bản 2 (GPLv2) là phiên bản 0.12.

Nhân Linux

Như đã đề cập ở trên, tất cả các bản phân phối Linux đều chạy cùng một hệ điều hành là Linux.

Bản Phân phối Linux

Khi mọi người nói về Red Hat Linux hoặc Ubuntu Linux thì tức là họ đang đề cập đến *bản phân phối Linux*. Bản phân phối Linux sẽ đi kèm với một nhân Linux và một môi trường làm cho nhân đó trở nên hữu ích theo cách mà chúng ta có thể tương tác với nó. Ở mức tối thiểu, chúng ta sẽ cần một vỏ dòng lệnh như Bash và một tập hợp các lệnh cơ bản cho phép chúng ta truy cập và quản lý hệ thống. Thông thường, tất nhiên các bản phân phối Linux sẽ có một Môi trường Máy tính để bàn đầy đủ như Gnome hoặc KDE.

Mặc dù mỗi bản phân phối Linux đều sẽ chạy hệ điều hành Linux, nhưng mỗi bản phân phối có thể sẽ khác nhau tùy theo phiên bản của hệ điều hành được sử dụng, tức *phiên bản của Nhân Linux đang được sử dụng* khi bản phân phối khởi động.

TIP

Nếu có quyền truy cập vào dòng lệnh Linux ngay vào lúc này, bạn có thể dễ dàng

kiểm tra phiên bản nhân Linux mà bạn đang chạy bằng cách đọc *bản phát hành hạt nhân*:

```
$ uname -r  
4.15.0-1019-aws
```

Các loại Bản Phân phối Linux

Việc chạy phiên bản mới nhất của nhân Linux có vẻ là một lựa chọn hiển nhiên, nhưng sự việc lại không hoàn toàn đơn giản như vậy. Chúng ta có thể phân loại một cách tương đối các bản phân phối Linux thành ba nhóm như sau:

- Bản phân phối Linux cấp Doanh nghiệp
 - Red Hat Enterprise Linux
 - CentOS
 - SUSE Linux Enterprise Server
 - Debian GNU/Linux
 - Ubuntu LTS
- Bản phân phối Linux cấp Người tiêu dùng
 - Fedora
 - Ubuntu non-LTS
 - openSUSE
- Bản phân phối Linux thử nghiệm và Hacker
 - Arch
 - Gentoo

Tất nhiên, đây chỉ là một tập hợp con rất nhỏ của các bản phân phối sẵn có, nhưng điều quan trọng ở đây là sự khác biệt giữa các bản phân phối cấp *doanh nghiệp*, *người tiêu dùng* và *thử nghiệm*, và lý do tại sao mỗi bản phân phối đó tồn tại.

Linux cấp Doanh nghiệp

Các bản phân phối như CentOS (*Community Enterprise OS*) được thiết kế để triển khai trong các tổ chức lớn sử dụng phần cứng doanh nghiệp. Nhu cầu của các doanh nghiệp rất khác so với nhu cầu của các đơn vị kinh doanh nhỏ lẻ, người sử dụng vì sở thích hoặc người dùng tại gia. Để đảm bảo tính khả dụng của các dịch vụ của mình, người dùng doanh nghiệp sẽ có yêu cầu

cao hơn về tính ổn định của phần cứng và phần mềm. Do đó, các bản phân phối Linux dành cho doanh nghiệp thường có xu hướng bao gồm các bản phát hành cũ hơn của hạt nhân và các phần mềm khác đã có tiếng là có thể hoạt động một cách đáng tin cậy. Thông thường, các bản phân phối sẽ chuyển các cập nhật quan trọng như sửa lỗi bảo mật trở lại các phiên bản ổn định này. Đối lại, các bản phân phối Linux dành cho doanh nghiệp có thể sẽ không hỗ trợ các phần cứng cấp người tiêu dùng mới nhất và chỉ cung cấp các phiên bản gói phần mềm cũ hơn. Tuy nhiên, giống như các bản phân phối Linux cấp người tiêu dùng, các doanh nghiệp cũng có xu hướng chọn các thành phần phần cứng trưởng thành và xây dựng dịch vụ của họ trên các phiên bản phần mềm ổn định.

Linux cấp Người tiêu dùng

Các bản phân phối như Ubuntu được nhắm tới các cơ sở kinh doanh nhỏ hoặc người dùng tại gia và người dùng vì sở thích. Do đó, họ cũng có khả năng sử dụng các phần cứng mới nhất được tìm thấy trên các hệ thống cấp người tiêu dùng. Để tối đa hóa hiệu suất của các phần cứng mới, những hệ thống này sẽ yêu cầu các trình điều khiển mới nhất, nhưng sự ổn định và độ tin cậy của cả phần cứng và trình điều khiển dường như không đáp ứng được nhu cầu của các doanh nghiệp lớn. Tuy nhiên, đối với thị trường tiêu dùng, hạt nhân mới nhất mới chính là thứ cần thiết với các trình điều khiển hiện đại nhất ngay cả khi chúng chưa được thử nghiệm nhiều. Nhân Linux mới sẽ có trình điều khiển mới nhất để hỗ trợ các phần cứng mới nhất có khả năng được sử dụng. Đặc biệt với sự phát triển mà chúng ta có thể thấy của Linux trên thị trường trò chơi điện tử, điều tối quan trọng là các trình điều khiển mới nhất phải có sẵn cho những người dùng này.

NOTE

Một số bản phân phối như Ubuntu có cung cấp cả phiên bản dành cho người tiêu dùng có chứa phần mềm mới và nhận các bản cập nhật trong một khoảng thời gian tương đối ngắn, cũng như cả phiên bản Hỗ trợ dài hạn (viết tắt là LTS - Long Term Support) phù hợp hơn với môi trường doanh nghiệp.

Bản phân phối Linux thử nghiệm và Hacker

Các bản phân phối như Arch Linux hoặc Gentoo Linux tồn tại trên đỉnh cao của công nghệ. Chúng chứa các phiên bản phần mềm mới nhất ngay cả khi các phiên bản này vẫn còn lỗi và các tính năng của chúng chưa được kiểm tra. Đối lại, các bản phân phối này có xu hướng sử dụng mô hình phát hành liên tục cho phép chúng cung cấp các bản cập nhật bất cứ lúc nào. Những bản phân phối này được sử dụng bởi những người dùng cao cấp luôn muốn có được các phần mềm mới nhất; họ biết rằng các chức năng có thể bị hỏng bất cứ lúc nào và có thể tự sửa chữa hệ thống của họ trong những trường hợp như vậy.

Nói tóm lại, khi đã chọn Linux làm hệ điều hành, nếu bạn đang sử dụng phần cứng cấp doanh nghiệp trên máy chủ hoặc máy tính để bàn thì bạn có thể tận dụng cả bản phân phối Linux cấp doanh nghiệp lẫn cấp tiêu dùng. Nếu đang sử dụng phần cứng dành cho người tiêu dùng và cần tận dụng tối đa những cải tiến phần cứng mới nhất thì có thể bạn sẽ cần một bản phân phối Linux

tương tự để phù hợp với nhu cầu của phần cứng.

Một số bản phân phối Linux có liên quan đến nhau. Ví dụ như Ubuntu được dựa trên Debian Linux và sử dụng cùng một hệ thống đóng gói là DPKG. Một ví dụ khác là Fedora - một bản thử nghiệm cho RedHat Enterprise Linux - nơi các tính năng tiềm năng của các phiên bản RHEL trong tương lai có thể được khám phá trước cả khi chúng có mặt trong bản phân phối doanh nghiệp.

Còn có nhiều bản phân phối Linux khác ngoài các bản phân phối mà chúng ta đã đề cập ở trên. Một lợi thế đi kèm với việc Linux là phần mềm mã nguồn mở là nhiều người có thể phát triển Linux theo tầm nhìn của riêng họ. Bằng cách đó, chúng ta có thể có hàng trăm bản phân phối. Để xem thêm các bản phân phối Linux, bạn có thể truy cập vào [Trang web của Distro Watch](#); những người quản lý trang web đã liệt kê 100 bản tải xuống hàng đầu của các bản phân phối Linux, cho phép bạn so sánh và xem xem bản nào hiện đang phổ biến.

Vòng đời Hỗ trợ của Linux

Có thể bạn đã nghĩ tới việc các bản phân phối Linux dành cho doanh nghiệp có vòng đời hỗ trợ lâu hơn so với các phiên bản Linux dành cho người tiêu dùng hoặc cộng đồng. Ví dụ như Red Hat Enterprise Linux có thời gian hỗ trợ là trong 10 năm. Red Hat Enterprise Linux 8 đã được ra mắt vào tháng 5 năm 2019, trong khi các bản cập nhật và hỗ trợ phần mềm sẽ có sẵn cho đến tháng 5 năm 2029.

Phiên bản dành cho người tiêu dùng thường sẽ chỉ có hỗ trợ cộng đồng thông qua các diễn đàn. Các bản cập nhật phần mềm thường có sẵn cho 3 bản phát hành. Nếu chúng ta lấy Ubuntu làm ví dụ thì tại thời điểm viết bài này, 19.04 là phiên bản mới nhất có sẵn các bản cập nhật cho tới khi phát hành 19.10 và sẽ dừng vào tháng 1 năm 2020. Ubuntu cũng cung cấp các phiên bản có hỗ trợ dài hạn được gọi là phiên bản *LTS*; chúng có thời gian hỗ trợ là 5 năm kể từ bản phát hành ban đầu. Phiên bản LTS hiện tại là 18.04 sẽ có các bản cập nhật phần mềm cho đến năm 2023. Các phiên bản LTS này biến Ubuntu trở thành một lựa chọn khả thi cho doanh nghiệp với sự hỗ trợ thương mại có sẵn từ Canonical (công ty đứng sau thương hiệu Ubuntu) hoặc các công ty tư vấn độc lập.

NOTE

Các bản phân phối Ubuntu sử dụng số phiên bản dựa trên ngày ở định dạng NN.TT: Ví dụ: phiên bản 19.04 được phát hành vào tháng 4 năm 2019.

Linux trong Máy tính để bàn của bạn

Sử dụng Linux làm hệ thống máy tính để bàn có thể khiến bạn gặp nhiều khó khăn hơn trong môi trường doanh nghiệp nơi việc hỗ trợ máy tính để bàn tập trung vào các dịch vụ hệ điều hành thương mại. Tuy nhiên, thách thức không chỉ có ở vấn đề hỗ trợ. Một khách hàng doanh nghiệp cũng có thể đã đầu tư lớn vào các giải pháp phần mềm gắn kết họ với các hệ điều hành máy tính

để bàn cụ thể. Như đã nói, có rất nhiều ví dụ về máy tính để bàn Linux được tích hợp vào các tổ chức lớn với các công ty như Amazon - họ thậm chí còn có bản phân phối Linux của riêng mình là [Amazon Linux 2](#). Bản phân phối này được sử dụng trên nền tảng đám mây AWS và cả trong nội bộ cho máy chủ và máy tính để bàn của họ.

Việc sử dụng Linux trong một cơ sở kinh doanh nhỏ hoặc tại gia sẽ trở nên dễ dàng hơn rất nhiều và có thể là một trải nghiệm bổ ích, loại bỏ được nhu cầu cấp phép và giúp bạn mở rộng tầm mắt với vô số phần mềm mã nguồn mở và tự do có sẵn cho Linux. Bạn cũng sẽ thấy rằng có rất nhiều môi trường máy tính để bàn khác nhau có sẵn. Phổ biến nhất là GNOME và KDE; ngoài ra vẫn có rất nhiều các lựa chọn khác. Quyết định sẽ phụ thuộc vào sở thích cá nhân của từng đối tượng.

Sử dụng Linux trên Máy chủ

Sử dụng Linux làm hệ điều hành máy chủ là việc đã trở nên rất phổ biến đối với cấp doanh nghiệp. Máy chủ sẽ được duy trì bởi các kỹ sư chuyên về Linux. Vì vậy, ngay cả khi có hàng nghìn người dùng thì những người dùng này vẫn có thể không hề biết gì về các máy chủ mà họ đang kết nối. Hệ điều hành máy chủ không quan trọng đối với họ và nói chung, các ứng dụng máy khách sẽ không quá khác nhau giữa Linux và các hệ điều hành khác trong phần phụ trợ. Cũng đúng là khi càng nhiều ứng dụng được ảo hóa hoặc chứa trong các đám mây cục bộ và từ xa thì hệ điều hành càng bị che lấp nhiều hơn và hệ điều hành nhúng khả năng cao sẽ là Linux.

Linux trong Đám mây

Một cơ hội khác để làm quen với Linux là triển khai Linux trên một trong nhiều đám mây công cộng có sẵn. Việc tạo tài khoản với một trong nhiều nhà cung cấp đám mây khác sẽ cho phép bạn nhanh chóng triển khai nhiều bản phân phối Linux khác nhau một cách nhanh chóng và dễ dàng.

Hệ điều hành không phải Linux

Dù là rất khó tin nhưng đúng là có những hệ điều hành không hề dựa trên nhân Linux. Trong những năm qua, tất nhiên đã có rất nhiều những hệ điều hành như vậy và cũng không ít đã thất bại, nhưng vẫn có những lựa chọn khác dành cho bạn dù là ở nhà hay trong văn phòng.

Unix

Trước khi Linux trở thành một hệ điều hành phổ biến thì ta có Unix. Unix từng được bán cùng với phần cứng và ngày nay vẫn còn một số bản Unix thương mại như AIX và HP-UX có sẵn trên thị trường. Trong khi Linux được lấy cảm hứng từ Unix (và tính thiếu khả dụng của nó đối với một số phần cứng nhất định) thì dòng hệ điều hành BSD được dựa trực tiếp trên Unix. Ngày nay, FreeBSD, NetBSD và OpenBSD cùng với một số hệ thống BSD có liên quan khác đều có sẵn dưới dạng phần mềm tự do.

Unix đã được sử dụng một cách rộng rãi trong doanh nghiệp, nhưng chúng ta đã được chứng kiến một sự suy giảm rõ rệt của Unix song song với sự phát triển của Linux. Khi Linux ngày càng phổ biến và các dịch vụ hỗ trợ doanh nghiệp cũng ngày càng được mở rộng, chúng ta đã thấy Unix dần dần biến mất. Solaris (ban đầu trực thuộc Sun trước khi chuyển sang Oracle) gần đây cũng đã biến mất. Đây là một trong những hệ điều hành Unix lớn được sử dụng bởi các công ty viễn thông và từ đó gọi với cái tên *Telco Grade Unix*.

Hệ điều hành Unix bao gồm:

- AIX
- FreeBSD, NetBSD, OpenBSD
- HP-UX
- Irix
- Solaris

macOS

macOS (trước đây là OS X) của Apple có từ năm 2001. Được dựa nhiều trên BSD Unix và sử dụng vỏ dòng lệnh Bash, đây là một hệ thống khá thân thiện để sử dụng nếu bạn đã quen sử dụng hệ điều hành Unix hoặc Linux. Nếu đang sử dụng macOS, bạn có thể mở ứng dụng cửa sổ dòng lệnh và chạy lại lệnh `uname`, chúng ta sẽ có thể kiểm tra được hệ điều hành được báo cáo:

```
$ uname -s  
Darwin
```

NOTE

Chúng ta có thể tận dụng tùy chọn `-s` trong trường hợp này để trả về tên hệ điều hành. Trước đó chúng ta đã sử dụng `-r` để trả về số phiên bản của hạt nhân.

Microsoft Windows

Vẫn có thể nói rằng phần lớn máy tính để bàn và máy tính xách tay đều sẽ chạy trên Windows. Hệ điều hành này đã thực sự thành công và thống trị thị trường máy tính để bàn trong nhiều năm. Mặc dù nó là phần mềm độc quyền và không miễn phí, nhưng thường thì khi người dùng mua phần cứng, giấy phép hệ điều hành sẽ được đi kèm nên việc này sẽ trở thành một lựa chọn dễ dàng. Tất nhiên, Windows được hỗ trợ rộng rãi bởi các nhà cung cấp phần cứng và phần mềm, cũng như có rất nhiều ứng dụng mã nguồn mở có sẵn cho nó. Tương lai của Windows dường như không tươi sáng như trước đây. Với số lượng máy tính để bàn và máy tính xách tay được bán ra ít hơn, thị trường hiện nay đang tập trung vào máy tính bảng và điện thoại. Nhưng phân khúc này đã bị Apple và Android thống trị nên Microsoft khó có thể giành được chỗ đứng.

Là một nền tảng máy chủ, Microsoft hiện cho phép khách hàng của mình lựa chọn giữa GUI (*Giao diện người dùng đồ họa*) và phiên bản chỉ dành cho dòng lệnh. Sự tách biệt giữa GUI và dòng lệnh là một điều quan trọng. Đa phần GUI của các Máy chủ Microsoft cũ sẽ được tải nhưng sẽ không có ai sử dụng nó. Hãy xem xét Bộ điều khiển miền Active Directory... người dùng luôn sử dụng nó để xác thực miền, nhưng nó lại được quản lý từ xa bởi máy tính để bàn của quản trị viên chứ không phải máy chủ.

Bài tập Hướng dẫn

1. Dự án nào tạo nên thành phần chung của tất cả các bản phân phối Linux?

CentOS	
Red Hat	
Ubuntu	
Linux Kernel	
CoreOS	

2. Hệ điều hành nào được báo cáo là đang sử dụng cho macOS của Apple?

OS X	
OSX	
Darwin	
MacOS	

3. Bản phân phối Linux khác với nhân Linux như thế nào?

Hạt nhân là một phần của bản phân phối, bản phân phối dưới dạng các ứng dụng bao quanh nhân để làm cho nó trở nên hữu ích	
Hạt nhân là bản phân phối Linux	
Tất cả các bản phân phối sử dụng cùng một nhân đều giống nhau	

4. Lựa chọn nào sau đây là một môi trường máy tính để bàn trong Linux?

Mint	
Elementary	
Zorin	
Gnome	

5. Thành phần nào của hệ điều hành cho phép truy cập vào phần cứng?

Trình điều khiển	
Trình Vở	
Dịch vụ	
Ứng dụng	

Bài tập Mở rộng

1. Hãy truy xuất bản phát hành Hạt nhân hiện tại của hệ thống Linux của bạn nếu bạn có quyền truy cập vào dòng lệnh.

2. Hãy sử dụng công cụ tìm kiếm ưa thích của bạn để định vị và xác định các nhà cung cấp đám mây công cộng có sẵn. Chúng có thể bao gồm AWS, Google Cloud, Rackspace, v.v. Hãy chọn một và xem hệ điều hành nào có sẵn để triển khai.

Tóm tắt

Trong bài này, bạn đã học cách phân biệt giữa các hệ điều hành khác nhau thường có sẵn. Chúng ta đã thảo luận về:

- Hệ điều hành dựa trên Linux
- UNIX
- Hệ điều hành Mac
- Hệ điều hành dựa trên Windows

Trong danh mục của Linux, chúng ta có thể chia nhỏ lựa chọn thành các bản phân phối có hỗ trợ dài hạn và những bản có chu kỳ hỗ trợ ngắn hơn. Các phiên bản LTS sẽ phù hợp hơn với Doanh nghiệp và các phiên bản hỗ trợ ngắn hạn hơn được nhắm đến người dùng tại gia và người dùng vì sở thích.

- Bản phân phối Linux cấp Doanh nghiệp
 - Red Hat Enterprise Linux
 - CentOS
 - SUSE Linux Enterprise Server
 - Debian GNU/Linux
 - Ubuntu LTS
- Bản phân phối Linux cấp Người tiêu dùng
 - Fedora
 - Ubuntu non-LTS
 - openSUSE
- Bản phân phối Linux thử nghiệm và Hacker
 - Arch
 - Gentoo

Đáp án Bài tập Hướng dẫn

1. Dự án nào tạo nên thành phần chung của tất cả các bản phân phối Linux?

CentOS	
Red Hat	
Ubuntu	
Linux Kernel	X
CoreOS	

2. Hệ điều hành nào được báo cáo là đang sử dụng cho macOS của Apple?

OS X	
OSX	
Darwin	X
MacOS	

3. Bản phân phối Linux khác với nhân Linux như thế nào?

Hạt nhân là một phần của bản phân phối, bản phân phối dưới dạng các ứng dụng bao quanh nhân để làm cho nó trở nên hữu ích	X
Hạt nhân là bản phân phối Linux	
Tất cả các bản phân phối sử dụng cùng một nhân đều giống nhau	

4. Lựa chọn nào sau đây là một môi trường máy tính để bàn trong Linux?

Mint	
Elementary	
Zorin	
Gnome	X

5. Thành phần nào của hệ điều hành cho phép truy cập vào phần cứng?

Trình điều khiển	X
Trình Vở	
Dịch vụ	
Ứng dụng	

Đáp án Bài tập Mở rộng

1. Hãy truy xuất bản phát hành Hạt nhân hiện tại của hệ thống Linux của bạn nếu bạn có quyền truy cập vào dòng lệnh.

```
$ uname -r  
4.15.0-47-generic
```

2. Hãy sử dụng công cụ tìm kiếm ưa thích của bạn để định vị và xác định các nhà cung cấp đám mây công cộng có sẵn. Chúng có thể bao gồm AWS, Google Cloud, Rackspace, v.v. Hãy chọn một và xem hệ điều hành nào có sẵn để triển khai.

Ví dụ, AWS sẽ cho phép bạn triển khai nhiều bản phân phối Linux như Debian, Red Hat, SUSE hoặc Ubuntu cũng như Windows.



**Linux
Professional
Institute**

4.2 Hiểu phần cứng của Máy tính

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 4.2](#)

Khối lượng

2

Các lĩnh vực kiến thức chính

- Phần cứng

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

- Bo mạch chủ, bộ xử lý, nguồn điện, ổ đĩa quang, thiết bị ngoại vi
- Ổ đĩa cứng, ổ đĩa trạng thái rắn và phân vùng, /dev/sd*
- Trình điều khiển



4.2 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	4 Hệ điều hành Linux
Mục tiêu:	4.2 Hiểu về Phần Cứng của Máy tính
Bài:	1 trên 1

Giới thiệu

Không có phần cứng, phần mềm sẽ chỉ giống như một tài liệu văn bản đơn thuần. Phần cứng sẽ xử lý các lệnh được mô tả bởi phần mềm và cung cấp các cơ chế lưu trữ, nhập và xuất. Suy cho cùng, ngay cả đám mây cũng là được hỗ trợ bởi phần cứng.

Là một hệ điều hành, một trong những trách nhiệm của Linux là cung cấp các phần mềm có giao diện để truy cập vào phần cứng của hệ thống. Hầu hết các chi tiết cụ thể về cấu hình đều nằm ngoài phạm vi của bài học này. Tuy nhiên, người dùng thường quan tâm đến hiệu suất, dung lượng và các yếu tố khác của phần cứng hệ thống vì chúng ảnh hưởng đến khả năng hỗ trợ tương xứng cho các ứng dụng cụ thể của hệ thống. Bài học này sẽ bàn về phần cứng dưới dạng các hạng mục vật lý riêng biệt sử dụng các đầu nối và giao diện tiêu chuẩn. Các tiêu chuẩn thường là tương đối cố định. Tuy nhiên, các yếu tố hình thức, hiệu suất và hiệu suất của phần cứng luôn không ngừng phát triển. Bất kể việc những thay đổi đó có thể làm lu mờ những sự khác biệt về mặt vật lý ra sao thì các khía cạnh khái niệm của phần cứng được mô tả trong bài học này vẫn có thể áp dụng được.

NOTE

Nhiều phần trong bài học này sẽ có chứa các ví dụ về dòng lệnh được sử dụng để minh họa các cách truy cập thông tin về phần cứng. Hầu hết các ví dụ là từ

Raspberry Pi B+, nhưng chúng cũng có thể áp dụng cho hầu hết các hệ thống. Tuy nhiên chúng ta hông nhất thiết phải hiểu các lệnh đó để có thể nắm được bài học này.

Nguồn Điện

Tất cả các thành phần hoạt động trong một hệ thống máy tính đều cần có điện để hoạt động. Thật không may, hầu hết các nguồn điện đều sẽ không phù hợp. Phần cứng của hệ thống máy tính yêu cầu một chỉ số điện áp cụ thể với dung sai tương đối chặt chẽ. Những ổ cắm cục bộ thường có trên tường của bạn lại thường không có sẵn những yếu tố này.

Những nguồn cung cấp năng lượng sẽ chuẩn hóa các nguồn năng lượng có sẵn. Các yêu cầu về điện áp tiêu chuẩn sẽ cho phép các nhà sản xuất tạo ra các thành phần phần cứng có thể được sử dụng trong các hệ thống ở mọi nơi trên thế giới. Bộ nguồn máy tính để bàn có xu hướng lấy điện từ các ổ cắm trên tường làm nguồn. Bộ nguồn máy chủ có xu hướng quan trọng hơn nên chúng thường có thể kết nối với nhiều nguồn để đảm bảo rằng chúng có thể tiếp tục hoạt động trong trường hợp có nguồn gặp sự cố.

Việc tiêu thụ điện năng sẽ tạo ra nhiệt. Nhiệt độ quá cao có thể khiến các bộ phận của hệ thống hoạt động chậm hoặc thậm chí gây ra hỏng hóc. Hầu hết các hệ thống đều có một số dạng quạt để tản nhiệt nhằm làm mát hiệu quả hơn. Các thành phần như bộ xử lý thường tạo ra nhiệt mà luồng không khí bình thường không thể làm tiêu tan hết được. Các bộ phận dễ nóng này thường có gắn các lá tản nhiệt đặc biệt được gọi là bộ tản nhiệt để giúp làm mát hiệu quả. Các bộ tản nhiệt thường có quạt cục bộ nhỏ riêng để đảm bảo luồng không khí tương thích.

Bo mạch chủ

Tất cả phần cứng của hệ thống cần phải được kết nối với nhau. Một bo mạch chủ sẽ chuẩn hóa kết nối đó bằng cách sử dụng các đầu nối và kích thước được tiêu chuẩn hóa. Nó cũng sẽ hỗ trợ cho cấu hình và nhu cầu điện năng của các đầu nối đó.

Có rất nhiều loại cấu hình bo mạch chủ khác nhau; chúng hỗ trợ các bộ xử lý và hệ thống bộ nhớ khác nhau. Bo mạch chủ thường được thiết kế để hỗ trợ nhiều loại đầu nối tiêu chuẩn hóa khác nhau. Các đầu nối này thường được thiết kế với kích cỡ và hình dáng khác nhau để tương thích với các vật chứa của chúng. Ngoại trừ khả năng kết nối các thiết bị bên ngoài cụ thể, cấu hình bo mạch chủ thực sự khá là rõ ràng đối với người dùng. Quản trị viên chủ yếu tiếp xúc với cấu hình bo mạch chủ khi có nhu cầu xác định các thiết bị cụ thể.

Khi nguồn được kết nối lần đầu tiên, phần cứng đặc thù dành cho bo mạch chủ phải được cấu hình và khởi tạo trước khi hệ thống có thể hoạt động. Bo mạch chủ sử dụng chương trình được lưu trữ trong bộ nhớ cố định được gọi là phần sụn để xử lý phần cứng cụ thể của bo mạch chủ. Hình

thức ban đầu của phần sụn bo mạch chủ được gọi là BIOS (*Hệ thống Nhập Xuất Cơ bản* - Basic Input/Output System). Ngoài các cài đặt cấu hình cơ bản, BIOS chủ yếu chịu trách nhiệm xác định, tải và chuyển hoạt động sang một hệ điều hành như Linux. Khi phần cứng phát triển, phần sụn cũng sẽ được mở rộng để hỗ trợ các ổ đĩa lớn hơn, các chẩn đoán, giao diện đồ họa, kết nối mạng và các khả năng nâng cao khác độc lập với bất kỳ hệ điều hành nào được tải. Những nỗ lực ban đầu để nâng cao phần sụn vượt ra khỏi BIOS cơ bản thường là việc riêng của các nhà sản xuất bo mạch chủ. Intel đã xác lập một tiêu chuẩn cho phần sụn nâng cao được gọi là EFI (*Giao diện phần sụn mở rộng* - Extensible Firmware Interface). Intel đã đóng góp EFI cho một tổ chức tiêu chuẩn để tạo ra UEFI (*Giao diện Phần mềm Mở rộng Hợp nhất* - Unified Extensible Firmware Interface). Ngày nay, hầu hết các bo mạch chủ đều sử dụng UEFI. BIOS và EFI hầu như không còn có mặt trên các hệ thống gần đây. Bất chấp điều này, hầu hết mọi người vẫn gọi phần sụn của bo mạch chủ là BIOS.

Có rất ít cài đặt phần sụn được người dùng phổ thông quan tâm, vì vậy chỉ những cá nhân chịu trách nhiệm về việc cấu hình phần cứng hệ thống mới cần làm việc với phần sụn và các cài đặt của nó. Một trong số ít các tùy chọn thường được thay đổi là việc kích hoạt các phần mở rộng ảo hóa của CPU hiện đại.

Bộ nhớ

Bộ nhớ hệ thống chứa dữ liệu và mã chương trình của các ứng dụng hiện đang chạy. Khi nói về bộ nhớ máy tính, hầu hết mọi người đều đề cập đến bộ nhớ hệ thống này. Một thuật ngữ phổ biến khác được sử dụng cho bộ nhớ hệ thống là từ viết tắt RAM (Random Access Memory - *Bộ nhớ Truy cập Ngẫu nhiên*) hoặc một số biến thể của từ viết tắt này. Đôi khi các tham chiếu đến vỏ bọc vật lý của bộ nhớ hệ thống như DIMM, SIMM hoặc DDR cũng được sử dụng.

Về mặt vật lý, bộ nhớ hệ thống thường được đóng gói trên các mô-đun bảng mạch riêng lẻ cắm vào bo mạch chủ. Các mô-đun bộ nhớ riêng lẻ hiện có kích thước từ 2 GB đến 64 GB. Đối với hầu hết các ứng dụng có mục đích phổ thông, 4 GB là bộ nhớ hệ thống tối thiểu mà mọi người nên cân nhắc. Đối với các máy trạm riêng lẻ, 16 GB thường đã quá đủ. Tuy nhiên, ngay cả 16 GB cũng có thể sẽ là hạn chế đối với người dùng đang chạy các ứng dụng trò chơi, video hoặc âm thanh cao cấp. Máy chủ thường yêu cầu bộ nhớ 128 GB hoặc thậm chí 256 GB để hỗ trợ tải lượng của người dùng một cách hiệu quả.

Nhìn chung, Linux cho phép người dùng coi bộ nhớ hệ thống như một chiếc hộp đen. Khi một ứng dụng được khởi động, Linux sẽ đảm nhận việc phân bổ bộ nhớ hệ thống cần thiết. Linux sẽ giải phóng bộ nhớ để các ứng dụng khác sử dụng khi một ứng dụng đã hoàn thành. Nhưng nếu một ứng dụng yêu cầu nhiều bộ nhớ hệ thống khả dụng hơn thì sao? Trong trường hợp này, Linux sẽ di chuyển các ứng dụng nhàn rỗi từ bộ nhớ hệ thống vào một vùng đĩa đặc biệt được gọi là không gian trao đổi. Linux sẽ di chuyển các ứng dụng nhàn rỗi từ không gian trao đổi đĩa trở lại bộ nhớ hệ thống khi chúng cần chạy.

Các hệ thống không có phần cứng video chuyên dụng thường sử dụng một phần bộ nhớ hệ thống (thường là 1 GB) để làm bộ lưu trữ hiển thị video. Điều này có thể làm giảm bộ nhớ hệ thống hiệu quả. Phần cứng video chuyên dụng thường có bộ nhớ riêng không khả dụng dưới dạng bộ nhớ hệ thống.

Có một số cách để lấy thông tin về bộ nhớ hệ thống. Với tư cách là người dùng, các giá trị ta thường quan tâm là tổng dung lượng bộ nhớ khả dụng và đang được sử dụng. Một trong số các nguồn để lấy thông tin là lệnh `free` cùng với tham số `-m` để sử dụng megabyte ở đầu ra:

```
$ free -m
              total        used          free      shared  buff/cache   available
Mem:           748           37           51          14          660          645
Swap:           99             0            99
```

Dòng đầu tiên biết tổng bộ nhớ khả dụng cho hệ thống (`total`), bộ nhớ đang sử dụng (`used`) và bộ nhớ trống (`free`). Dòng thứ hai hiển thị các thông tin này về không gian trao đổi. Các bộ nhớ được gọi là `shared` và `buff/cache` thì hiện đang được sử dụng cho các chức năng khác của hệ thống dù dung lượng được chỉ định trong `available` (có sẵn) có thể được sử dụng cho ứng dụng.

Bộ Vi Xử lý

Thuật ngữ “bộ vi xử lý” ám chỉ một thứ gì đó đang được xử lý. Trong máy tính, hầu hết quá trình xử lý đó là xử lý các tín hiệu điện. Thông thường, các tín hiệu đó được coi là có một trong hai giá trị nhị phân 1 hoặc 0.

Khi mọi người nói về máy tính, họ thường sử dụng cụm từ bộ vi xử lý thay thế cho từ viết tắt CPU (*Bộ xử lý Trung tâm* - Central Processing Unit). Điều này không đúng về mặt kỹ thuật. Mọi máy tính có mục đích sử dụng phổ thông đều có CPU xử lý các lệnh nhị phân do phần mềm chỉ định. Vì vậy, việc mọi người cho rằng bộ vi xử lý và CPU là một cũng khá dễ hiểu. Tuy nhiên, ngoài CPU, các máy tính hiện đại thường bao gồm các bộ vi xử lý dành riêng cho tác vụ khác. Có lẽ bộ vi xử lý bổ sung dễ nhận biết nhất là GPU (*Bộ xử lý đồ họa* - Graphical Processing Unit). Do đó, dù CPU đúng là một bộ xử lý thì không phải bộ vi xử lý nào cũng là CPU.

Đối với hầu hết mọi người, kiến trúc CPU là một tham chiếu đến các lệnh mà bộ xử lý hỗ trợ. Mặc dù Intel và AMD đều tạo ra các bộ xử lý hỗ trợ các tập lệnh giống nhau nhưng việc phân biệt theo nhà cung cấp cũng là cần thiết do sự khác biệt về ngoại hình, hiệu suất và mức tiêu thụ điện. Các bản phân phối phần mềm thường sử dụng các ký hiệu sau để chỉ định tập lệnh tối thiểu mà chúng cần để có thể vận hành:

i386

Tham khảo từ tập lệnh 32 bit được liên kết với Intel 80386.

x86

Thường tham khảo các tập lệnh 32 bit được liên kết với các phiên bản kế tiếp của 80386, chẳng hạn như 80486, 80586 và Pentium.

x64/x86-64

Tham khảo bộ xử lý hỗ trợ cả tập lệnh 32-bit và 64-bit của dòng x86.

AMD

Tham chiếu đến phần hỗ trợ x86 của bộ xử lý AMD.

AMD64

Tham chiếu đến hỗ trợ x64 của bộ xử lý AMD.

ARM

Tham chiếu một CPU của *Máy tính có Tập lệnh rút gọn* (RISC - Reduced Instruction Set Computer) không dựa trên tập lệnh x86. Nó thường được sử dụng bởi các thiết bị nhúng, di động, máy tính bảng và các thiết bị chạy bằng pin. Có một phiên bản của Linux dành cho ARM được Raspberry Pi sử dụng.

Tệp `/proc/cpuinfo` chứa thông tin chi tiết về bộ xử lý của hệ thống. Thật không may, các chi tiết này lại không được thân thiện với người dùng phổ thông. Ta có thể đạt được một kết quả tổng quát hơn bằng lệnh `lscpu`. Đầu ra từ Raspberry Pi B+:

```
$ lscpu
Architecture:          armv7l
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:  0-3
Thread(s) per core:   1
Core(s) per socket:   4
Socket(s):             1
Model:                 4
Model name:            ARMv7 Processor rev 4 (v7l)
CPU max MHz:           1400.0000
CPU min MHz:           600.0000
BogoMIPS:              38.40
Flags:                 half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32
                       lpae evtstrm crc32
```

Đối với hầu hết mọi người, việc có vô số các nhà cung cấp, các dòng bộ xử lý và các yếu tố thông số kỹ thuật đại diện cho một loạt các lựa chọn gây hoang mang. Dù vậy, có một số yếu tố liên quan đến CPU và bộ xử lý mà ngay cả người dùng phổ thông và quản trị viên cũng cần phải xem xét khi họ cần chỉ định các môi trường hoạt động:

Kích thước bit

Đối với CPU, con số này liên quan đến cả kích thước gốc của dữ liệu mà nó phải thao tác và cả dung lượng bộ nhớ mà nó có thể truy cập. Hầu hết các hệ thống hiện đại đều là 32-bit hoặc 64-bit. Nếu một ứng dụng cần hơn 4 gigabyte bộ nhớ thì ứng dụng đó phải chạy trên hệ thống 64 bit vì 4 gigabyte là mốc tối đa có thể chạy được bằng 32 bit. Và trong khi các ứng dụng 32-bit thường có thể chạy trên các hệ thống 64-bit, các ứng dụng 64-bit lại không thể chạy trên các hệ thống 32-bit.

Tốc độ đồng hồ

Thường được biểu thị bằng megahertz (MHz) hoặc gigahertz (GHz). Yếu tố này liên quan đến tốc độ xử lý lệnh của bộ xử lý. Nhưng tốc độ của bộ xử lý chỉ là một trong những yếu tố ảnh hưởng đến thời gian phản hồi, thời gian chờ và thông lượng của hệ thống. Ngay cả một người dùng đa tác vụ cũng hiếm khi để cho CPU của một máy tính để bàn thông thường hoạt động nhiều hơn 2 hoặc 3 phần trăm thời gian. Bất kể bạn có thường xuyên sử dụng các ứng dụng nặng liên quan đến các hoạt động như mã hóa hoặc kết xuất video hay không thì tốc độ CPU vẫn có thể tác động đáng kể đến thông lượng và thời gian chờ.

Bộ nhớ đệm

CPU yêu cầu một luồng liên tục của cả lệnh và dữ liệu để hoạt động. Chi phí và mức tiêu thụ năng lượng của bộ nhớ hệ thống nhiều gigabyte có thể được truy cập ở tốc độ xung nhịp của CPU khá là cao. Bộ nhớ đệm tốc độ CPU được tích hợp vào chip CPU để cung cấp bộ đệm tốc độ cao giữa CPU và bộ nhớ hệ thống. Bộ nhớ đệm được chia thành nhiều lớp, thường được gọi là L1, L2, L3 và thậm chí cả L4. Thông thường, có càng nhiều bộ nhớ đệm thì hệ thống hoạt động càng hiệu quả.

Lõi

Lõi dùng để chỉ một CPU riêng lẻ. Ngoài lõi đại diện cho một CPU vật lý, *Công nghệ siêu phân luồng* (HTT - Hyper-Threading Technology) cũng cho phép một CPU vật lý duy nhất xử lý đồng thời nhiều lệnh, do đó nó hoạt động gần như tương đương với nhiều CPU vật lý. Trong đa số các trường hợp, nhiều lõi vật lý sẽ được đóng gói dưới dạng một chip xử lý vật lý duy nhất. Tuy nhiên, có những bo mạch chủ có thể hỗ trợ nhiều chip xử lý vật lý. Trên lý thuyết, khi có nhiều lõi để xử lý các tác vụ hơn thì thông lượng hệ thống sẽ tốt hơn. Thật không may, các ứng dụng dành cho máy tính để bàn thường chỉ khiến CPU bận 2 hoặc 3 phần trăm thời gian; do đó, việc thêm nhiều CPU nhân rồi sẽ không mấy cải thiện được thông lượng. Việc sử dụng nhiều lõi sẽ thích hợp nhất để chạy các ứng dụng được viết để có nhiều luồng hoạt động độc lập, chẳng hạn

như kết xuất khung hình video, kết xuất trang web hoặc môi trường máy ảo đa người dùng.

Lưu trữ

Các thiết bị lưu trữ cung cấp một phương pháp để giữ lại các chương trình và dữ liệu. Ổ ổ cứng (HDD - Hard Disk Drives) và Ổ đĩa thể rắn (SSD - Solid State Drives) là các dạng thiết bị lưu trữ phổ biến nhất cho máy chủ và máy tính để bàn. Thẻ nhớ USB và các thiết bị quang học như DVD cũng được sử dụng nhưng chúng hiếm khi được dùng làm thiết bị chính.

Đúng như tên gọi, ổ cứng lưu trữ thông tin trên một hoặc nhiều ổ cứng vật lý. Các đĩa vật lý sẽ được phủ bằng phương tiện từ tính để có thể lưu trữ. Các đĩa được chứa trong một gói kín vì bụi, hạt nhỏ và thậm chí cả dấu vân tay cũng có thể cản trở khả năng đọc và ghi phương tiện từ tính của ổ cứng.

SSD là phiên bản phức tạp hơn của thẻ nhớ USB với dung lượng lớn hơn đáng kể. SSD lưu trữ thông tin trong vi mạch nên không có bộ phận chuyển động.

Mặc dù các công nghệ cơ bản của ổ cứng HDD và SSD là khác nhau, nhưng giữa chúng vẫn có những yếu tố quan trọng có thể so sánh được. Dung lượng của ổ HDD dựa trên quy mô của các thành phần vật lý, trong khi dung lượng của ổ SSD lại phụ thuộc vào số lượng vi mạch. Trên mỗi gigabyte, ổ SSD tốn gấp từ 3 đến 10 lần so với ổ HDD. Để đọc hoặc ghi, ổ HDD phải đợi một vị trí trên đĩa xoay đến một vị trí đã biết, trong khi ổ SSD có thể truy cập ngẫu nhiên. Tốc độ truy cập của SSD thường nhanh hơn từ 3 đến 5 lần so với thiết bị HDD. Vì không có bộ phận chuyển động nên SSD sẽ tiêu thụ ít năng lượng hơn và đáng tin cậy hơn so với HDD.

Dung lượng lưu trữ vẫn không ngừng tăng lên đối với ổ cứng HDD và SSD. Ngày nay, ổ HDD 5 terabyte và ổ SSD 1 terabyte là những tiêu chuẩn tương đối phổ biến. Dù vậy, dung lượng lưu trữ lớn hơn không phải lúc nào cũng là tốt hơn. Khi một thiết bị lưu trữ bị lỗi, thông tin chứa trong đó sẽ không còn khả dụng nữa. Và tất nhiên, việc sao lưu sẽ mất nhiều thời gian hơn khi có nhiều thông tin cần sao lưu. Đối với các ứng dụng đọc và ghi nhiều dữ liệu, độ trễ và hiệu suất có thể sẽ quan trọng hơn dung lượng.

Các hệ thống hiện đại sử dụng SCSI (*Giao diện Hệ thống Máy tính nhỏ* - Small Computer System Interface) hoặc SATA (*Phần đính kèm Công nghệ nâng cao Nối tiếp* - Serial AT Attachment) để kết nối với các thiết bị lưu trữ. Các giao diện này thường được hỗ trợ bởi đầu nối thích hợp trên bo mạch chủ. Lướt tải ban đầu sẽ đến từ một thiết bị lưu trữ được gắn vào bo mạch chủ. Việc cài đặt chương trình cơ sở sẽ xác định thứ tự truy cập thiết bị cho lần tải đầu tiên này.

Các hệ thống lưu trữ được gọi là RAID (*Hệ thống Đĩa dự phòng* - Redundant Array of Independent Disks) và là cách triển khai phổ biến để tránh mất thông tin. Một mảng RAID bao gồm nhiều thiết bị vật lý chứa các bản sao thông tin trùng lặp. Nếu một thiết bị bị lỗi, tất cả các thông tin vẫn sẽ có

sẵn. Các cấu hình RAID vật lý khác nhau được tham chiếu là 0, 1, 5, 6 và 10. Mỗi tên gọi có kích thước lưu trữ, đặc điểm hiệu suất và cách lưu trữ dữ liệu dư thừa hoặc tổng kiểm tra để phục hồi dữ liệu khác nhau. Ngoài những tác vụ bổ sung về cấu hình quản trị, RAID tương đối dễ nắm bắt.

Các thiết bị lưu trữ thường đọc và ghi dữ liệu dưới dạng các khối byte. Lệnh `lsblk` có thể được sử dụng để liệt kê các thiết bị khối có sẵn cho hệ thống. Ví dụ sau được chạy trên Raspberry Pi sử dụng thẻ SD làm thiết bị lưu trữ. Thông tin chi tiết về đầu ra sẽ được đề cập trong các bài học về *Phân vùng* và *Trình điều khiển* tiếp theo:

```
$ lsblk
NAME          MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk0       179:0    0 29.7G  0 disk
+-mmcblk0p1   179:1    0 43.9M  0 part /boot
+-mmcblk0p2   179:2    0 29.7G  0 part /
```

Phân vùng

Một thiết bị lưu trữ thực chất là một chuỗi dài các vị trí lưu trữ. Phân vùng là cơ chế cho Linux biết nó nên xem các vị trí lưu trữ này là một chuỗi đơn lẻ hay nhiều chuỗi độc lập. Mỗi phân vùng sẽ được xử lý như thể nó là một thiết bị riêng lẻ. Hầu hết các phân vùng sẽ được tạo khi hệ thống được cấu hình lần đầu tiên. Nếu cần thay đổi, các công cụ quản trị sẽ có sẵn để quản lý việc phân vùng thiết bị.

Vậy tại sao yếu tố nhiều phân vùng lại được ưa chuộng? Một số ví dụ về việc sử dụng phân vùng là quản lý dung lượng lưu trữ khả dụng, cô lập thời gian mã hoá bổ sung hoặc hỗ trợ nhiều hệ thống tệp. Các phân vùng sẽ cho phép việc chỉ có một thiết bị lưu trữ duy nhất có thể khởi động trong các hệ điều hành khác nhau.

Mặc dù Linux có thể nhận diện trình tự lưu trữ của thiết bị thô, thiết bị thô lại không thể được sử dụng nguyên trạng. Để sử dụng một thiết bị thô, thiết bị đó phải được định dạng. Việc định dạng sẽ ghi một hệ thống tệp vào một thiết bị và chuẩn bị hệ thống này cho các hoạt động của tệp. Không có hệ thống tệp, thiết bị sẽ không thể được sử dụng cho các hoạt động liên quan đến tệp.

Người dùng xem các phân vùng như thể chúng là các thiết bị riêng lẻ. Điều này khiến bạn dễ dàng bỏ qua một thực tế là bạn vẫn đang xử lý một thiết bị vật lý duy nhất. Cụ thể, các hoạt động từ thiết bị đến thiết bị (trên thực tế là hoạt động của phân vùng đến phân vùng) sẽ không có hiệu suất như mong đợi. Một thiết bị duy nhất là một cơ chế vật lý với một bộ phần cứng đọc/ghi. Quan trọng hơn, bạn không thể sử dụng các phân vùng của một thiết bị vật lý đơn lẻ như một thiết kế chịu lỗi. Nếu thiết bị bị lỗi, tất cả các phân vùng đều sẽ bị lỗi, vậy nên khả năng chịu lỗi là không có.

NOTE

Trình quản lý Khối lượng hợp lý (LVM - Logical Volume Manager) là một khả năng của phần mềm cho phép quản trị viên kết hợp các đĩa với phân vùng đĩa riêng lẻ và xử lý chúng như thể chúng là một ổ đĩa đơn.

Thiết bị ngoại vi

Máy chủ và máy trạm cần sự kết hợp của CPU, bộ nhớ hệ thống và bộ lưu trữ để hoạt động. Nhưng những thành phần cơ bản này sẽ không giao tiếp trực tiếp với thế giới bên ngoài. Thiết bị ngoại vi là thiết bị cung cấp cho hệ thống đầu vào, đầu ra và khả năng tương tác với phần còn lại của thế giới thực.

Hầu hết các bo mạch chủ đều có các đầu nối bên ngoài tích hợp sẵn và hỗ trợ phần sụn cho các giao diện ngoại vi kế thừa phổ biến hỗ trợ các thiết bị như bàn phím, chuột, âm thanh, video và mạng. Gần đây, các bo mạch chủ thường có đầu nối Ethernet để hỗ trợ mạng, đầu nối HDMI hỗ trợ các nhu cầu đồ họa cơ bản và một hoặc nhiều cổng USB (*Universal Serial Bus*) cho hầu hết những thứ khác. Có nhiều phiên bản của USB với các đặc tính vật lý và tốc độ khác nhau. Việc có nhiều phiên bản cổng USB trên một bo mạch chủ là khá phổ biến.

Bo mạch chủ cũng có thể có một hoặc nhiều khe cắm mở rộng. Các khe cắm mở rộng cho phép người dùng thêm các bảng mạch đặc biệt được gọi là thẻ mở rộng hỗ trợ các thiết bị ngoại vi tùy chỉnh, kế thừa và không theo tiêu chuẩn. Giao diện đồ họa, âm thanh và mạng là những thẻ mở rộng phổ biến. Thẻ mở rộng cũng hỗ trợ RAID và các giao diện kế thừa định dạng đặc biệt liên quan đến kết nối nối tiếp và song song.

Cấu hình *Hệ thống trên một Vi mạch (SoC - System on a Chip)* đạt được các lợi thế về sức mạnh, hiệu suất, không gian và độ tin cậy so với cấu hình bo mạch chủ thông qua cách đóng gói bộ xử lý, bộ nhớ hệ thống, SSD và phần cứng để điều khiển các thiết bị ngoại vi dưới dạng một gói mạch tích hợp duy nhất. Các thiết bị ngoại vi được cấu hình SoC hỗ trợ sẽ bị giới hạn bởi các thành phần trong gói. Do đó, các cấu hình SoC có xu hướng được phát triển cho các mục đích sử dụng cụ thể. Điện thoại, máy tính bảng và các thiết bị di động khác thường được dựa trên công nghệ SoC.

Một số hệ thống cũng có kết hợp các thiết bị ngoại vi. Máy tính xách tay cũng tương tự như máy trạm nhưng nó có tích hợp các thiết bị ngoại vi màn hình, bàn phím và chuột mặc định. Hệ thống tất cả-trong-một cũng tương tự như máy tính xách tay nhưng chúng yêu cầu thiết bị ngoại vi là chuột và bàn phím. Bộ điều khiển dựa trên bo mạch chủ hoặc SoC thường được đính kèm với các thiết bị ngoại vi tích hợp sẵn phù hợp với mục đích sử dụng cụ thể.

Trình Điều khiển và Tệp Thiết bị

Đến đây, bài học đã trình bày các thông tin về bộ xử lý, bộ nhớ, đĩa, phân vùng, định dạng và thiết bị ngoại vi. Tuy vậy, việc yêu cầu người dùng phổ thông xử lý các chi tiết cụ thể cho từng thiết bị

trong hệ thống của họ sẽ khiến những hệ thống đó trở nên không sử dụng được. Tương tự, các nhà phát triển phần mềm sẽ cần sửa đổi mã của họ cho mọi thiết bị mới hoặc thiết bị đã qua sửa đổi mà họ cần hỗ trợ.

Giải pháp cho các vấn đề thuộc phạm trù “chi tiết” này là trình điều khiển thiết bị. Trình điều khiển thiết bị chấp nhận một bộ yêu cầu tiêu chuẩn, sau đó sẽ dịch các yêu cầu đó thành các hoạt động điều khiển thích hợp với thiết bị. Trình điều khiển thiết bị là thứ cho phép bạn và các ứng dụng bạn chạy đọc từ tệp `/home/carol/stuff` mà không cần lo lắng liệu tệp đó có nằm trên ổ cứng, ổ cứng thể rắn, thẻ nhớ, bộ lưu trữ được mã hóa hay một số thiết bị khác hay không.

Tệp thiết bị được tìm thấy trong thư mục `/dev` và sẽ xác định các thiết bị vật lý, quyền truy cập thiết bị và các trình điều khiển được hỗ trợ. Theo quy ước, trong các hệ thống hiện đại sử dụng thiết bị lưu trữ dựa trên SCSI hoặc SATA, tên tệp đặc tả sẽ bắt đầu bằng tiền tố `sd`. Tiếp theo tiền tố sẽ là một chữ cái (chẳng hạn như `a` hoặc `b`) để cho biết đó là một thiết bị vật lý. Sau tiền tố và số nhận dạng thiết bị sẽ là một số chỉ ra một phân vùng trong thiết bị vật lý. Vì vậy, `/dev/sda` sẽ tham chiếu toàn bộ thiết bị lưu trữ đầu tiên, trong khi `/dev/sda3` sẽ tham chiếu phân vùng 3 trong thiết bị lưu trữ đầu tiên. Tệp thiết bị cho từng loại thiết bị có quy ước đặt tên phù hợp với mỗi thiết bị. Mặc dù việc đề cập đến tất cả các quy ước đặt tên có thể nằm ngoài phạm vi của bài học này, nhưng điều quan trọng cần nhớ ở đây là các quy ước này rất quan trọng trong việc giúp tác vụ quản trị hệ thống trở nên khả thi.

Mặc dù nội dung của thư mục `/dev` nằm ngoài phạm vi của bài học này, việc xem xét đầu vào cho thiết bị lưu trữ lại khá là hữu ích. Các tệp thiết bị dành cho thẻ SD thường sử dụng `mmcblk` làm tiền tố:

```
$ ls -l mmcblk*
brw-rw---- 1 root disk 179, 0 Jun 30 01:17 mmcblk0
brw-rw---- 1 root disk 179, 1 Jun 30 01:17 mmcblk0p1
brw-rw---- 1 root disk 179, 2 Jun 30 01:17 mmcblk0p2
```

Các chi tiết liệt kê cho một tệp thiết bị khác với các chi tiết tệp thông thường:

- Không giống như tệp hoặc thư mục, chữ cái đầu tiên của trường quyền là `b`. Điều này cho biết rằng các khối được đọc và ghi vào thiết bị theo khối thay vì các ký tự riêng lẻ.
- Trường kích thước là hai giá trị được phân tách bằng dấu phẩy thay vì một giá trị. Giá trị đầu tiên thường chỉ ra một trình điều khiển cụ thể trong hạt nhân và giá trị thứ hai sẽ chỉ định một thiết bị cụ thể được điều khiển bởi trình điều khiển.
- Tên tệp sử dụng một chữ số cho thiết bị vật lý để quy ước đặt tên có thể thích nghi bằng cách chỉ định hậu tố phân vùng là `p`, theo sau là một chữ số.

Mỗi thiết bị hệ thống phải có một đầu vào trong `/dev`. Vì nội dung của thư mục `/dev` được tạo khi cài đặt nên thường ta sẽ có các đầu vào cho mọi trình điều khiển và thiết bị có thể có ngay cả khi không có thiết bị vật lý nào tồn tại.

Bài tập Hướng dẫn

1. Hãy mô tả các thuật ngữ sau đây:

Bộ vi xử lý	
CPU	
GPU	

2. Nếu bạn chủ yếu chạy các ứng dụng chỉnh sửa video (một hoạt động tính toán cường độ cao), bạn cho rằng các thành phần và đặc điểm nào sẽ có tác động nhiều nhất đến khả năng sử dụng hệ thống:

Lõi CPU	
Tốc độ CPU	
Bộ nhớ hệ thống khả dụng	
Hệ thống lưu trữ	
GPU	
Trình hiển thị video	
Không có đáp án nào đúng	

3. Bạn nghĩ tên của tệp thiết bị trong `/dev` sẽ là gì cho phân vùng 3 của ổ đĩa SATA thứ ba trong một hệ thống:

<code>sd3p3</code>	
<code>sdcp3</code>	
<code>sdc3</code>	
Không có đáp án nào đúng	

Bài tập Mở rộng

1. Hãy chạy lệnh `lsblk` trên hệ thống của bạn. Hãy xác định các tham số dưới đây. Nếu hệ thống không khả dụng ngay lập tức, hãy xem danh sách `lsblk -f` dành cho hệ thống Raspberry Pi được đề cập trong phần “Lưu trữ” ở trên:

```
$ lsblk -f
NAME          FSTYPE LABEL  UUID                                 MOUNTPOINT
mmcblk0
+-mmcblk0p1  vfat   boot   9304-D9FD                           /boot
+-mmcblk0p2  ext4   rootfs 29075e46-f0d4-44e2-a9e7-55ac02d6e6cc /
```

- Loại thiết bị và số lượng
- Cấu trúc phân vùng của từng thiết bị
- Loại hệ thống tệp và gắn kết cho mỗi phân vùng

Tóm tắt

Một hệ thống là một tập hợp tổng của các thành phần của nó. Các thành phần khác nhau tác động đến chi phí, hiệu suất và khả năng sử dụng theo những cách khác nhau. Mặc dù có các loại cấu hình phổ biến cho máy trạm và máy chủ nhưng lại không có một cấu hình duy nhất nào là tốt nhất.

Đáp án Bài tập Hướng dẫn

1. Hãy mô tả các thuật ngữ sau đây:

Bộ xử lý

Một thuật ngữ chung áp dụng cho bất kỳ loại bộ xử lý nào. Thường được sử dụng một cách không chính xác như một từ đồng nghĩa với CPU.

CPU

Bộ xử lý trung tâm. Một đơn vị xử lý cung cấp hỗ trợ cho các tác vụ tính toán có mục đích phổ thông.

GPU

Bộ xử lý đồ họa. Một đơn vị xử lý được tối ưu hóa để hỗ trợ các hoạt động liên quan đến việc trình bày đồ họa.

2. Nếu bạn chủ yếu chạy các ứng dụng chỉnh sửa video (một hoạt động tính toán cường độ cao), bạn cho rằng các thành phần và đặc điểm nào sẽ có tác động nhiều nhất đến khả năng sử dụng hệ thống:

Lỗi CPU

Có. Việc có nhiều lõi sẽ hỗ trợ các tác vụ hiển thị và kết xuất đồng thời theo yêu cầu của tác vụ chỉnh sửa video.

Tốc độ CPU

Có. Việc kết xuất video cần một lượng lớn các hoạt động tính toán.

Bộ nhớ hệ thống khả dụng

Có khả năng. Video không được nén được sử dụng trong chỉnh sửa sẽ có dung lượng lớn. Các hệ thống có mục đích phổ thông thường đi kèm với 8 gigabyte bộ nhớ. Bộ nhớ với 16 hoặc thậm chí 32 gigabyte sẽ cho phép hệ thống xử lý nhiều khung hình hơn của video không nén, giúp các hoạt động chỉnh sửa trở nên hiệu quả hơn.

Hệ thống lưu trữ

Có. Các tệp video thường có dung lượng khá lớn. Việc sử dụng ổ đĩa SSD cục bộ sẽ giúp việc truyền tải trở nên hiệu quả hơn. Ổ đĩa mạng chậm hơn có khả năng sẽ phản tác dụng.

GPU

Không. GPU chủ yếu tác động đến việc trình bày video được kết xuất.

Trình hiển thị video

Không. Trình hiển thị video chủ yếu ảnh hưởng đến việc trình bày video được hiển thị.

Không có đáp án nào đúng

Không. Một số yếu tố trên có tác động rõ ràng đến mức độ khả dụng của hệ thống của bạn.

3. Bạn nghĩ tên của tệp thiết bị trong `/dev` sẽ là gì cho phân vùng 3 của ổ đĩa SATA thứ ba trong một hệ thống:

<code>sd3p3</code>	Không đúng. Ổ 3 sẽ là <code>sdc</code> , không phải <code>sd3</code>
<code>sdcp3</code>	Không đúng. Phân vùng 3 sẽ là 3, không phải <code>p3</code>
<code>sdc3</code>	Đúng
Không có đáp án nào đúng	Không đúng. Câu trả lời đúng nằm trong số các đáp án trên.

Đáp án Bài tập Mở rộng

- Hãy chạy lệnh `lsblk` trên hệ thống của bạn. Hãy xác định các tham số dưới đây. Nếu hệ thống không khả dụng ngay lập tức, hãy xem danh sách `lsblk -f` dành cho hệ thống Raspberry Pi được đề cập trong phần “Lưu trữ” ở trên:

```
$ lsblk -f
NAME          FSTYPE LABEL  UUID                                 MOUNTPOINT
mmcblk0
+-mmcblk0p1  vfat   boot   9304-D9FD                           /boot
+-mmcblk0p2  ext4   rootfs 29075e46-f0d4-44e2-a9e7-55ac02d6e6cc /
```

Các câu trả lời sau đây được dựa trên danh sách `lsblk -f` cho hệ thống Raspberry Pi ở trên. Có thể có nhiều cách trả lời:

Loại thiết bị và số lượng

Có một thiết bị: `mmcblk0`. Theo quy ước, ta biết rằng `mmcblk` sẽ là thẻ nhớ SD.

Cấu trúc phân vùng của từng thiết bị

Có hai phân vùng: `mmcblk0p1` và `mmcblk0p2`.

Loại hệ thống tệp và gắn kết cho từng phân vùng

Phân vùng 1 sử dụng hệ thống tệp `vfat`. Nó được sử dụng để khởi động hệ thống và được gắn là `/boot`. Phân vùng 2 sử dụng hệ thống tệp `ext4`. Nó được sử dụng làm hệ thống tệp chính và được gắn là `/`.



4.3 Nơi lưu trữ dữ liệu

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 4.3](#)

Khối lượng

3

Các lĩnh vực kiến thức chính

- Chương trình và cấu hình
- Quy trình
- Địa chỉ bộ nhớ
- Tin nhắn hệ thống
- Ghi nhật ký

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

- `ps`, `top`, `free`
- `syslog`, `dmesg`
- `/etc/`, `/var/log/`
- `/boot/`, `/proc/`, `/dev/`, `/sys/`



4.3 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	4 Hệ điều hành Linux
Mục tiêu:	4.3 Nơi lưu trữ Dữ liệu
Bài:	1 trên 2

Giới thiệu

Đối với một hệ điều hành, mọi thứ đều được coi là dữ liệu. Đối với Linux, tất cả mọi thứ đều được coi là tệp: chương trình, tệp thông thường, thư mục, thiết bị khối (ổ cứng, v.v.), thiết bị kiểu ký tự (bảng điều khiển, v.v.), các quy trình của hạt nhân, ổ cắm, phân vùng, liên kết, v.v. Cấu trúc của thư mục trong Linux được bắt đầu từ gốc (root) /; đây là một tập hợp các tệp chứa dữ liệu. Việc mọi thứ đều là một tệp chính là một tính năng mạnh mẽ của Linux vì điều này cho phép ta tinh chỉnh hầu như mọi khía cạnh của hệ thống.

Trong bài học này, chúng ta sẽ thảo luận về các vị trí khác nhau mà trong đó dữ liệu quan trọng được lưu trữ theo thiết lập Tiêu chuẩn Phân cấp Hệ thống Tệp Linux (FHS). Một trong số các vị trí này là các thư mục thực lưu trữ dữ liệu liên tục trên đĩa, trong khi các vị trí khác lại là hệ thống tệp giả được tải trong bộ nhớ và cho phép chúng ta truy cập vào dữ liệu hệ thống con của hạt nhân như các quy trình đang chạy, việc sử dụng bộ nhớ, cấu hình phần cứng, v.v. Dữ liệu được lưu trữ trong các thư mục ảo này được sử dụng bởi một số lệnh cho phép chúng ta theo dõi và xử lý chúng.

Các Chương trình và Cấu hình của chúng

Các dữ liệu quan trọng trên hệ thống Linux — không cần phải tranh cãi — chính là các chương trình và tệp cấu hình của chúng. Các chương trình chính là các tệp thực thi chứa các tập lệnh được chạy bởi bộ xử lý của máy tính, trong khi các tệp cấu hình thường là những tài liệu văn bản điều khiển hoạt động của chương trình. Các tệp thực thi có thể là tệp nhị phân hoặc tệp văn bản. Các tệp văn bản có thể thực thi được gọi là tệp lệnh. Dữ liệu cấu hình trên Linux thường cũng được lưu trữ trong các tệp văn bản dù vẫn có nhiều cách biểu diễn dữ liệu cấu hình khác nữa.

Nơi lưu trữ Tệp Nhị phân

Giống như bất kỳ một tệp nào khác, các tệp thực thi sẽ nằm trong các thư mục từ `/`. Cụ thể hơn, các chương trình sẽ được phân phối theo cấu trúc ba tầng: tầng thứ nhất (`/`) bao gồm các chương trình có thể sẽ cần thiết cho chế độ một người dùng; tầng thứ hai (`/usr`) chứa hầu hết các chương trình đa người dùng; tầng thứ ba (`/usr/local`) được sử dụng để lưu trữ các phần mềm không được cung cấp bởi bản phân phối và đã được biên dịch cục bộ.

Các vị trí điển hình cho các chương trình bao gồm:

`/sbin`

Chứa các tệp nhị phân cần thiết để quản trị hệ thống, chẳng hạn như `parted` hoặc `ip`.

`/bin`

Chứa các tệp nhị phân cần thiết cho tất cả người dùng, chẳng hạn như `ls`, `mv` hoặc `mkdir`.

`/usr/sbin`

Lưu trữ các tệp nhị phân để quản trị hệ thống, chẳng hạn như `deluser` hoặc `groupadd`.

`/usr/bin`

Bao gồm hầu hết các tệp thực thi — chẳng hạn như `free`, `pstree`, `sudo` hoặc `man` — có thể được sử dụng bởi tất cả người dùng.

`/usr/local/sbin`

Được sử dụng để lưu trữ các chương trình được cài đặt cục bộ để quản trị hệ thống và không được quản lý bởi trình quản lý gói của hệ thống.

`/usr/local/bin`

Phục vụ mục đích tương tự như `/usr/local/sbin` nhưng dành cho các chương trình người dùng thông thường.

Gần đây, một số bản phân phối đã bắt đầu thay thế `/bin` và `/sbin` bằng các liên kết tượng trưng

đến `/usr/bin` và `/usr/sbin`.

NOTE

Thư mục `/opt` đôi khi cũng được sử dụng để lưu trữ các ứng dụng tùy chọn của bên thứ ba.

Ngoài các thư mục này, người dùng thông thường có thể có các chương trình của riêng họ trong một trong hai địa chỉ sau đây:

- `/home/$USER/bin`
- `/home/$USER/.local/bin`

TIP

Bạn có thể tìm ra những thư mục nào có sẵn để từ đó chạy các tệp nhị phân bằng cách tham chiếu biến `PATH` với `echo $PATH`. Để biết thêm thông tin về `PATH`, hãy xem lại các bài học về biến và tùy chỉnh vỏ.

Chúng ta có thể tìm thấy vị trí của các chương trình bằng lệnh `which`:

```
$ which git
/usr/bin/git
```

Nơi lưu trữ Tệp Cấu hình

Thư mục `/etc`

Trong những ngày đầu, Unix sẽ có thư mục riêng cho từng loại dữ liệu, chẳng hạn như `/bin` cho các tệp nhị phân và `/boot` cho hạt nhân. Tuy nhiên, `/etc` (có nghĩa là *vân vân*) được tạo dưới dạng thư mục tổng hợp để lưu trữ bất kỳ tệp nào không thuộc các danh mục trên. Hầu hết các tệp này đều là tệp cấu hình. Theo thời gian, ngày càng có nhiều tệp cấu hình đã được thêm vào nên `/etc` đã trở thành thư mục chính cho các tệp cấu hình của các chương trình. Như đã nói ở trên, một tệp cấu hình thường là một văn bản (trái ngược với tệp nhị phân) thuần túy cục bộ điều khiển hoạt động của một chương trình.

Trong `/etc`, chúng ta có thể tìm thấy các mẫu khác nhau cho tên tệp cấu hình:

- Các tệp có phần mở rộng *ad hoc* hoặc không có phần mở rộng nào, chẳng hạn như:

group

Cơ sở dữ liệu nhóm hệ thống.

hostname

Tên của máy chủ.

hosts

Danh sách địa chỉ IP và bản dịch tên máy chủ của chúng.

passwd

Cơ sở dữ liệu người dùng hệ thống—bao gồm bảy trường được phân tách bằng dấu hai chấm cung cấp thông tin về người dùng.

profile

Tập cấu hình toàn hệ thống cho Bash.

shadow

Tập được mã hóa cho mật khẩu người dùng.

- Các tập khởi tạo kết thúc bằng `rc`:

bash.bashrc

Tập `.bashrc` trên toàn hệ thống dành cho trình vỏ bash tương tác.

nanorc

Tập khởi tạo mẫu cho GNU nano (một trình soạn thảo văn bản đơn giản thường đi kèm với bất kỳ bản phân phối nào).

- Các tập kết thúc bằng `.conf`:

resolv.conf

Tập cấu hình cho trình phân giải—cung cấp quyền truy cập vào Hệ thống tên miền Internet (DNS - Domain Name System).

sysctl.conf

Tập cấu hình để đặt các biến hệ thống cho nhân.

- Các thư mục có đuôi `.d`:

Một số chương trình có một tập cấu hình duy nhất (`*.conf` hoặc khác) đã phát triển để có thư mục `*.d` chuyên dụng giúp xây dựng các cấu hình mô-đun mạnh mẽ hơn. Ví dụ: để định cấu hình logrotate, bạn sẽ tìm `logrotate.conf` cũng như các thư mục `logrotate.d`.

Cách tiếp cận này khá là hữu ích trong những trường hợp mà các ứng dụng khác nhau cần cấu hình cho cùng một dịch vụ cụ thể. Ví dụ: nếu một gói máy chủ web chứa cấu hình logrotate thì tức là cấu hình có thể hiện đang được đặt trong một tập chuyên dụng trong thư mục `logrotate.d`. Tập này có thể được cập nhật bởi gói máy chủ web mà không phải cần phải can

thiệt vào cấu hình logrotate còn lại. Tương tự như vậy, các gói có thể thêm các tác vụ cụ thể bằng cách đặt các tệp trong thư mục `/etc/cron.d` thay vì sửa đổi `/etc/crontab`.

Trong Debian — cũng như các phái sinh của nó — cách tiếp cận như vậy đã được áp dụng cho danh sách các nguồn đáng tin cậy được đọc bởi công cụ quản lý gói `apt`: ngoài `/etc/apt/sources.list` cổ điển, giờ đây chúng ta có thể tìm thư mục `/etc/apt/sources.list.d`:

```
$ ls /etc/apt/sources*
/etc/apt/sources.list
/etc/apt/sources.list.d:
```

Tệp cấu hình trong HOME (Dotfiles)

Ở cấp độ người dùng, các chương trình lưu trữ cấu hình và thiết lập của chúng trong các tệp ẩn trong thư mục chính của người dùng (cũng được biểu thị bằng `~`). Hãy nhớ rằng, các tệp ẩn sẽ được bắt đầu bằng dấu chấm (`.` - dot) — do đó tên của chúng là *dotfiles*.

Một số tệp dotfile này là các tệp lệnh Bash sẽ tùy chỉnh phiên vỏ của người dùng và được cung cấp ngay sau khi người dùng đăng nhập vào hệ thống:

.bash_history

Lưu trữ lịch sử dòng lệnh.

.bash_logout

Bao gồm các lệnh để thực thi khi rời khỏi vỏ đăng nhập.

.bashrc

Tệp lệnh khởi tạo của Bash cho vỏ không đăng nhập.

.profile

Tệp lệnh khởi tạo của Bash cho các vỏ đăng nhập.

NOTE

Tham khảo bài học về “Khái niệm cơ bản về Dòng lệnh” để tìm hiểu thêm về Bash và các tệp khởi tạo của nó.

Các tệp cấu hình của những chương trình dành riêng cho từng người dùng khác sẽ được lấy nguồn khi các chương trình tương ứng của chúng được khởi động: `.gitconfig`, `.emacs.d`, `.ssh`, v.v.

Nhân Linux

Trước khi bất kỳ quy trình nào có thể chạy, hạt nhân phải được tải vào một vùng bộ nhớ được bảo vệ. Sau đó, quy trình với PID 1 (ngày nay thường là `systemd`) sẽ khởi động chuỗi quy trình, tức là một quy trình sẽ bắt đầu (các) quy trình khác và cứ tiếp tục như vậy. Khi các quy trình đang hoạt động, nhân Linux sẽ chịu trách nhiệm phân bổ tài nguyên cho chúng (bàn phím, chuột, đĩa, bộ nhớ, giao diện mạng, v.v.).

NOTE

Trước `systemd`, `/sbin/init` luôn là quy trình chạy đầu tiên trong hệ thống Linux như một phần của trình quản lý hệ thống *System V Init*. Trên thực tế, hiện tại bạn vẫn có thể tìm thấy `/sbin/init` nhưng nó sẽ được liên kết với `/lib/systemd/systemd`.

Nơi lưu trữ Hạt nhân: `/boot`

Hạt nhân nằm trong `/boot` — cùng với các tệp liên quan đến tác vụ khởi động khác. Hầu hết các tệp này đều có các thành phần số phiên bản của hạt nhân trong tên của chúng (phiên bản hạt nhân, bản sửa đổi chính, bản sửa đổi phụ và số bản vá).

Thư mục `/boot` sẽ bao gồm các loại tệp sau với tên tương ứng với phiên bản của hạt nhân:

`config-4.9.0-9-amd64`

Cài đặt cấu hình cho hạt nhân, chẳng hạn như các tùy chọn và mô-đun được biên dịch cùng với hạt nhân.

`initrd.img-4.9.0-9-amd64`

Hình ảnh đĩa RAM ban đầu giúp ích trong quá trình khởi động bằng cách tải hệ thống tệp gốc tạm thời vào bộ nhớ.

`System-map-4.9.0-9-amd64`

Tệp `System-map` (trên một số hệ thống sẽ được đặt tên là `System.map`) có chứa các địa chỉ bộ nhớ cho các tên ký hiệu của hạt nhân. Mỗi khi hạt nhân được xây dựng lại, nội dung của tệp sẽ thay đổi do các địa chỉ bộ nhớ có thể sẽ khác nhau. Hạt nhân sẽ sử dụng tệp này để tra cứu các địa chỉ bộ nhớ cho một ký hiệu hạt nhân cụ thể hoặc ngược lại.

`vmlinuz-4.9.0-9-amd64`

Hạt nhân thích hợp ở định dạng tự giải nén, tiết kiệm không gian và nén (nên ta có `z` trong `vmlinuz`; `vm` là viết tắt của bộ nhớ ảo — `virtual memory` — và sẽ bắt đầu được sử dụng khi hạt nhân lần đầu tiên nhận được hỗ trợ cho bộ nhớ ảo).

grub

Thư mục cấu hình cho bộ nạp khởi động grub2.

TIP

Bởi là một tính năng quan trọng của hệ điều hành nên nhiều hơn một hạt nhân và các tệp liên quan của nó sẽ được giữ trong /boot phòng trường hợp hạt nhân mặc định bị lỗi và ta phải quay lại ít nhất là một phiên bản trước đó để có thể khởi động hệ thống và sửa chữa nó.

Thư mục /proc

Thư mục /proc là một trong những hệ thống tệp ảo hoặc hệ thống tệp giả vì nội dung của nó không được ghi vào đĩa mà sẽ được tải trong bộ nhớ. Nó sẽ được điền động mỗi khi máy tính khởi động và sẽ liên tục phản ánh trạng thái hiện tại của hệ thống. /proc sẽ bao gồm các thông tin về:

- Các quy trình đang chạy
- Cấu hình nhân
- Phần cứng hệ thống

Bên cạnh tất cả các dữ liệu liên quan đến các quy trình mà chúng ta sẽ thấy trong bài học tiếp theo, thư mục này cũng lưu trữ các tệp có thông tin về phần cứng của hệ thống và cài đặt cấu hình của hạt nhân. Một số tệp này sẽ bao gồm:

/proc/cpuinfo

Nó lưu trữ thông tin về CPU của hệ thống:

```
$ cat /proc/cpuinfo
processor      : 0
vendor_id    : GenuineIntel
cpu family   : 6
model        : 158
model name   : Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
stepping     : 10
cpu MHz      : 3696.000
cache size   : 12288 KB
(...)
```

/proc/cmdline

Nó lưu trữ các chuỗi được chuyển đến hạt nhân khi khởi động:

```
$ cat /proc/cmdline
```

```
BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro
quiet
```

/proc/modules

Nó hiển thị danh sách các mô-đun được tải vào hạt nhân:

```
$ cat /proc/modules
nls_utf8 16384 1 - Live 0xffffffffc0644000
isofs 40960 1 - Live 0xffffffffc0635000
udf 90112 0 - Live 0xffffffffc061e000
crc_itu_t 16384 1 udf, Live 0xffffffffc04be000
fuse 98304 3 - Live 0xffffffffc0605000
vboxsf 45056 0 - Live 0xffffffffc05f9000 (0)
joydev 20480 0 - Live 0xffffffffc056e000
vboxguest 327680 5 vboxsf, Live 0xffffffffc05a8000 (0)
hid_generic 16384 0 - Live 0xffffffffc0569000
(...)
```

Thư mục /proc/sys

Thư mục này bao gồm các thiết lập về cấu hình của hạt nhân trong các tệp được phân loại thành các danh mục trên mỗi thư mục con:

```
$ ls /proc/sys
abi debug dev fs kernel net user vm
```

Hầu hết các tệp này hoạt động giống như một công tắc và — do đó — chỉ chứa một trong hai giá trị: 0 hoặc 1 (“bật” hoặc “tắt”). Ví dụ:

/proc/sys/net/ipv4/ip_forward

Giá trị cho phép hoặc vô hiệu hóa máy của người dùng hoạt động như một bộ định tuyến (có thể chuyển tiếp các gói):

```
$ cat /proc/sys/net/ipv4/ip_forward
0
```

Tuy nhiên cũng có một số trường hợp ngoại lệ:

/proc/sys/kernel/pid_max

PID tối đa cho phép:

```
$ cat /proc/sys/kernel/pid_max
32768
```

WARNING

Hãy hết sức cẩn thận khi thay đổi cài đặt nhân vì giá trị sai có thể dẫn đến một hệ thống không ổn định.

Thiết bị Phần Cứng

Hãy nhớ rằng, trong Linux, “mọi thứ đều là tệp”. Điều này ngụ ý rằng thông tin thiết bị phần cứng cũng như cài đặt cấu hình riêng của hạt nhân đều sẽ được lưu trữ trong các tệp đặc biệt nằm trong các thư mục ảo.

Thư mục /dev

Thư mục *thiết bị* /dev chứa các tệp thiết bị (hoặc các nút) dành cho tất cả các thiết bị phần cứng được kết nối. Các tệp thiết bị này được sử dụng làm giao diện giữa các thiết bị và quy trình sử dụng chúng. Mỗi tệp thiết bị sẽ thuộc một trong hai loại sau:

Thiết bị khối

Là những dữ liệu được đọc và ghi trong các khối có thể được xử lý riêng lẻ. Ví dụ: ổ cứng (và các phân vùng của chúng, như /dev/sda1), ổ flash USB, CD, DVD, v.v.

Thiết bị kiểu ký tự: Là những dữ liệu mà trong đó dữ liệu được đọc và ghi tuần tự từng ký tự một. Các ví dụ có thể bao gồm bàn phím, bảng điều khiển văn bản (/dev/console), cổng nối tiếp (chẳng hạn như /dev/ttyS0, v.v.), v.v.

Khi liệt kê các tệp thiết bị, hãy đảm bảo rằng bạn có sử dụng `ls` với khóa chuyển `-l` để phân biệt giữa hai tệp. Ví dụ, chúng ta có thể kiểm tra ổ cứng và các phân vùng:

```
# ls -l /dev/sd*
brw-rw---- 1 root disk 8, 0 may 25 17:02 /dev/sda
brw-rw---- 1 root disk 8, 1 may 25 17:02 /dev/sda1
brw-rw---- 1 root disk 8, 2 may 25 17:02 /dev/sda2
(...)
```

Hoặc cho các chương trình cửa sổ dòng lệnh nối tiếp (TeleTYpewriter):

```
# ls -l /dev/tty*
crw-rw-rw- 1 root tty      5,  0 may 25 17:26 /dev/tty
crw--w---- 1 root tty      4,  0 may 25 17:26 /dev/tty0
crw--w---- 1 root tty      4,  1 may 25 17:26 /dev/tty1
(...)
```

Hãy lưu ý ký tự đầu tiên là **b** cho thiết bị khối (block devices) và **c** cho thiết bị ký tự (character devices).

TIP Dấu hoa thị (*) là một ký tự khớp mẫu khối ám chỉ 0 hoặc nhiều ký tự. Do đó mà nó khá quan trọng trong các lệnh `ls -l /dev/sd*` và `ls -l /dev/tty*` ở trên. Để tìm hiểu thêm về các ký tự đặc biệt này, hãy tham khảo bài học về khớp mẫu khối.

Hơn nữa, `/dev` bao gồm một số tệp đặc biệt khá hữu ích cho các mục đích lập trình khác nhau:

`/dev/zero`

Cung cấp ký tự rỗng theo yêu cầu.

`/dev/null`

Tức *bit bucket*. Nó loại bỏ tất cả các thông tin được gửi đến mình.

`/dev/urandom`

Tạo ra các số giả ngẫu nhiên.

Thư mục `/sys`

Hệ thống tệp `sys` (`sysfs`) được gắn với `/sys`. Nó được giới thiệu cùng lúc với sự ra đời của hạt nhân 2.6 và đại diện cho một cải tiến lớn trên `/proc/sys`.

Các quy trình cần tương tác với các thiết bị trong `/dev` và do đó, hạt nhân cần có một thư mục chứa thông tin về các thiết bị phần cứng này. Thư mục này là `/sys` và dữ liệu của nó sẽ được sắp xếp theo thứ tự thành các danh mục. Chẳng hạn, để kiểm tra địa chỉ MAC của thẻ mạng của bạn (`enp0s3`), bạn sẽ sử dụng lệnh `cat` với tệp sau:

```
$ cat /sys/class/net/enp0s3/address
08:00:27:02:b2:74
```

Bộ nhớ và các loại Bộ nhớ

Về cơ bản, để một chương trình bắt đầu chạy, nó phải được tải vào bộ nhớ. Nhìn chung, khi nói về

bộ nhớ tức là chúng ta đang đề cập đến *Bộ nhớ truy cập ngẫu nhiên* (RAM) và — khi so sánh với các ổ cứng cơ học — nó có ưu điểm là nhanh hơn rất nhiều. Tuy vậy, nó lại không được ổn định (tức là khi máy tính tắt, dữ liệu sẽ biến mất).

Mặc dù đã nói ở trên — khi nói đến bộ nhớ — chúng ta có thể phân biệt thành hai loại chính trong hệ thống Linux:

Bộ nhớ vật lý

Còn được gọi là *RAM*; nó có dạng chip được tạo thành từ các mạch tích hợp chứa hàng triệu bóng bán dẫn và tụ điện. Chúng sẽ tạo thành các ô nhớ (khối xây dựng cơ bản của bộ nhớ máy tính). Mỗi ô này có một mã thập lục phân đính kèm — tức một địa chỉ bộ nhớ — để nó có thể được tham chiếu khi cần.

Bộ nhớ Trao đổi

Còn được gọi là *không gian trao đổi*, đây là phần bộ nhớ ảo nằm trên ổ cứng và được sử dụng khi không còn RAM.

Mặt khác, ta có một khái niệm về *bộ nhớ ảo*: là sự trừu tượng hóa tổng dung lượng bộ nhớ có thể sử dụng được và có địa chỉ (RAM, cũng như dung lượng ổ đĩa) mà các ứng dụng có thể nhìn thấy.

Lệnh `free` sẽ phân tích cú pháp `/proc/meminfo` và hiển thị lượng bộ nhớ trống và đã sử dụng trong hệ thống một cách rất rõ ràng:

```
$ free
```

	total	used	free	shared	buff/cache	available
Mem:	4050960	1474960	1482260	96900	1093740	2246372
Swap:	4192252	0	4192252			

Hãy cùng diễn giải các cột này:

total

Tổng dung lượng bộ nhớ vật lý và trao đổi được cài đặt.

used

Dung lượng bộ nhớ vật lý và trao đổi hiện đang được sử dụng.

free

Dung lượng bộ nhớ vật lý và trao đổi hiện đang không được sử dụng.

shared

Dung lượng bộ nhớ vật lý được sử dụng — chủ yếu — bởi `tmpfs`.

buff/cache

Dung lượng bộ nhớ vật lý hiện đang được sử dụng bởi bộ đệm hạt nhân, các bộ nhớ mảnh và bộ đệm trang.

available

Ước tính dung lượng bộ nhớ vật lý có sẵn cho các quy trình mới.

Theo mặc định, lệnh `free` sẽ hiển thị các giá trị theo kibibyte, nhưng nó cũng cho phép nhiều tùy chọn hiển thị kết quả của nó theo các đơn vị đo lường khác nhau. Một số tùy chọn này bao gồm:

-b

Bytes.

-m

Mebibytes.

-g

Gibibytes.

-h

Định dạng con người có thể đọc được.

`-h` thì sẽ luôn luôn dễ đọc:

```
$ free -h
              total        used          free      shared  buff/cache   available
Mem:           3,9G         1,4G         1,5G          75M         1,0G         2,2G
Swap:          4,0G           0B          4,0G
```

NOTE

Một kibibyte (KiB) bằng 1.024 byte trong khi một kilobyte (KB) bằng 1000 byte. Tương tự như vậy với mebibytes, gibibytes, v.v.

Bài tập Hướng dẫn

1. Hãy sử dụng lệnh `which` để tìm ra vị trí của các chương trình sau và hoàn thành bảng:

Chương trình	Lệnh which	Đường dẫn đến Tập Thực thi (đầu ra)	Người dùng cần đặc quyền hệ thống?
swapon			
kill			
cut			
usermod			
cron			
ps			

2. Các tệp sau đây được tìm thấy ở đâu?

Tệp	/etc	~
.bashrc		
bash.bashrc		
passwd		
.profile		
resolv.conf		
sysctl.conf		

3. Hãy giải thích ý nghĩa của các phần tử số của tệp hạt nhân `vmlinuz-4.15.0-50-generic` được tìm thấy trong `/boot`:

Phần Tử Số	Ý nghĩa
4	
15	
0	
50	

4. Bạn sẽ sử dụng lệnh nào để liệt kê tất cả các ổ cứng và phân vùng trong `/dev`?

Bài tập Mở rộng

- Các tệp thiết bị của ổ cứng được trình bày dựa trên bộ điều khiển mà chúng sử dụng — chúng ta đã thấy `/dev/sd*` cho các ổ đĩa sử dụng SCSI (Giao diện Hệ thống Máy tính nhỏ) và SATA (Phần đính kèm Công nghệ Nâng cao Nối tiếp), nhưng
 - Các ổ đĩa IDE (Điện tử truyền động tích hợp) cũ được trình bày như thế nào?
 - Và Ổ đĩa NVMe (Bộ nhớ nhanh ổn định) hiện đại thì sao?
- Hãy xem tệp `/proc/meminfo`. Hãy so sánh nội dung của tệp này với đầu ra của lệnh `free` và xác định khóa nào từ `/proc/meminfo` sẽ tương ứng với các trường sau trong đầu ra của `free`:

đầu ra free	trường /proc/meminfo
total	
free	
shared	
buff/cache	
available	

Tóm tắt

Trong bài học này, bạn đã học về vị trí của các chương trình và các tệp cấu hình của chúng trong hệ thống Linux. Những điều quan trọng ta cần nhớ là:

- Về cơ bản, các chương trình có thể được tìm thấy trong cấu trúc thư mục ba cấp: `/`, `/usr` và `/usr/local`. Mỗi cấp độ này có thể chứa các thư mục `bin` và `sbin`.
- Các tệp cấu hình được lưu trữ trong `/etc` và `~`.
- Dotfiles là những tệp ẩn bắt đầu bằng dấu chấm (`.`).

Chúng ta cũng đã thảo luận về nhân Linux. Các vấn đề quan trọng cần nhớ là: * Đối với Linux, mọi thứ đều là tệp. * Nhân Linux nằm trong `/boot` cùng với các tệp liên quan đến khởi động khác. * Để các quy trình bắt đầu tiến hành, trước tiên hạt nhân phải được tải vào vùng bộ nhớ được bảo vệ. * Công việc của hạt nhân là phân bổ tài nguyên hệ thống cho các quy trình. * Hệ thống tệp ảo (hoặc giả) `/proc` sẽ lưu trữ dữ liệu hệ thống và hạt nhân quan trọng theo một cách không ổn định.

Tương tự như vậy, chúng ta cũng đã khám phá các thiết bị phần cứng và học được những điều sau:

- Thư mục `/dev` lưu trữ các tệp đặc biệt (còn gọi là các nút) cho tất cả các thiết bị phần cứng được kết nối: *thiết bị khối* hoặc *thiết bị kiểu ký tự*. Thiết bị khối truyền dữ liệu theo khối; thiết bị kiểu ký tự truyền từng ký tự một.
- Thư mục `/dev` cũng chứa các tệp đặc biệt khác như `/dev/zero`, `/dev/null` hoặc `/dev/urandom`.
- Thư mục `/sys` lưu trữ thông tin về các thiết bị phần cứng được sắp xếp thành các danh mục.

Cuối cùng, chúng ta đã tìm hiểu về bộ nhớ. Chúng ta đã học về:

- Một chương trình sẽ chạy khi nó được nạp vào bộ nhớ.
- RAM (Bộ nhớ truy cập ngẫu nhiên) là gì.
- Bộ nhớ trao đổi là gì.
- Cách hiển thị việc sử dụng bộ nhớ.

Các lệnh được dùng trong bài này:

cat

Nối/in nội dung tệp.

free

Hiển thị dung lượng bộ nhớ trống và đã sử dụng trong hệ thống.

ls

Liệt kê nội dung thư mục.

which

Hiển thị vị trí của chương trình.

Đáp án Bài tập Hướng dẫn

1. Hãy sử dụng lệnh `which` để tìm ra vị trí của các chương trình sau và hoàn thành bảng:

Chương trình	Lệnh which	Đường dẫn đến Tập Thực thi (đầu ra)	Người dùng cần đặc quyền hệ thống?
swapon	which swapon	/sbin/swapon	Có
kill	which kill	/bin/kill	Không
cut	which cut	/usr/bin/cut	Không
usermod	which usermod	/usr/sbin/usermod	Có
cron	which cron	/usr/sbin/cron	Có
ps	which ps	/bin/ps	Không

2. Các tệp sau đây được tìm thấy ở đâu?

Tệp	/etc	~
.bashrc	Không	Có
bash.bashrc	Có	Không
passwd	Có	Không
.profile	Không	Có
resolv.conf	Có	Không
sysctl.conf	Có	Không

3. Hãy giải thích ý nghĩa của các phần tử số của tệp hạt nhân `vmlinuz-4.15.0-50-generic` được tìm thấy trong `/boot`:

Phần Tử Số	Ý nghĩa
4	Phiên bản Hạt Nhân
15	Bản chỉnh sửa Chính
0	Bản chỉnh sửa Phụ
50	Số bản vá

4. Bạn sẽ sử dụng lệnh nào để liệt kê tất cả các ổ cứng và phân vùng trong `/dev`?

```
ls /dev/sd*
```

Đáp án Bài tập Mở rộng

1. Các tệp thiết bị của ổ cứng được trình bày dựa trên bộ điều khiển mà chúng sử dụng — chúng ta đã thấy `/dev/sd*` cho các ổ đĩa sử dụng SCSI (Giao diện Hệ thống Máy tính nhỏ) và SATA (Phần đính kèm Công nghệ Nâng cao Nối tiếp), nhưng

- Các ổ đĩa IDE (Điện tử truyền động tích hợp) cũ được trình bày như thế nào?

```
/dev/hd*
```

- Và Ổ đĩa NVMe (Bộ nhớ nhanh ổn định) hiện đại thì sao?

```
/dev/nvme*
```

2. Hãy xem tệp `/proc/meminfo`. Hãy so sánh nội dung của tệp này với đầu ra của lệnh `free` và xác định khóa nào từ `/proc/meminfo` tương ứng với các trường sau trong đầu ra của `free`:

đầu ra free	trường /proc/meminfo
total	MemTotal / SwapTotal
free	MemFree / SwapFree
shared	Shmem
buff/cache	Buffers, Cached và SReclaimable
available	MemAvailable



4.3 Bài 2

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	4 Hệ điều hành Linux
Mục tiêu:	4.3 Nơi lưu trữ Dữ liệu
Bài:	2 trên 2

Giới thiệu

Sau khi đã khám phá các chương trình và tệp cấu hình của chúng, chúng ta sẽ cùng tìm hiểu về cách các lệnh được thực thi dưới dạng các quy trình trong bài học này. Tương tự, chúng ta sẽ thảo luận về thông báo của hệ thống, việc sử dụng bộ đệm vòng của hạt nhân và những sự thay đổi so với trước đây mà `systemd` và `journald` (nhật ký chạy nền của `systemd`) đã đóng góp trong việc ghi nhật ký hệ thống.

Quy trình

Mỗi khi người dùng đưa ra một lệnh, một chương trình sẽ chạy và một hoặc nhiều quy trình sẽ được tạo ra.

Các quy trình tồn tại trong một hệ thống phân cấp. Sau khi hạt nhân được tải vào bộ nhớ khi khởi động, quy trình đầu tiên sẽ được bắt đầu; việc này sẽ kích hoạt các quy trình khác và các quy trình này cũng có thể lại tiếp tục kích hoạt các quy trình khác nữa. Mỗi quy trình có một mã định danh (PID) và mã định danh quy trình gốc (PPID) duy nhất. Đây là những số nguyên dương được gán theo thứ tự liên tiếp.

Khám phá các Quy trình Động: top

Bạn có thể nhận được danh sách động của tất cả các quy trình đang chạy bằng lệnh `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
 (...)
```

Như chúng ta đã thấy ở trên, lệnh `top` cũng có thể cung cấp cho chúng ta thông tin về mức tiêu thụ bộ nhớ và CPU của toàn bộ hệ thống cũng như của từng quy trình.

Lệnh `top` cũng cho phép người dùng thực hiện một số tương tác.

Theo mặc định, đầu ra sẽ được sắp xếp theo thứ tự giảm dần của tỷ lệ phần trăm thời gian CPU được sử dụng bởi mỗi quy trình. Hành vi này có thể được sửa đổi bằng cách nhấn các phím sau từ bên trong `top`:

M

Sắp xếp theo mức sử dụng bộ nhớ.

N

Sắp xếp theo số ID của quy trình.

T

Sắp xếp theo thời gian chạy.

P

Sắp xếp theo phần trăm sử dụng CPU.

Để chuyển đổi giữa thứ tự giảm dần/tăng dần, ta chỉ cần nhấn R.

TIP

Một phiên bản cao cấp và thân thiện với người dùng hơn của `top` là `htop`. Một cách khác — có lẽ toàn diện hơn — là `atop`. Nếu nó chưa được cài đặt trong hệ thống của bạn, bạn nên sử dụng trình quản lý gói của mình để cài đặt chúng và dùng thử.

Hình ảnh tức thời của Các Quy trình: `ps`

Một lệnh rất hữu ích khác dùng để lấy thông tin về các quy trình là `ps`. Trong khi `top` cung cấp thông tin động, `ps` lại cũng cấp các thông tin tĩnh.

Nếu ta gọi `ps` mà không có tùy chọn, đầu ra của nó sẽ khá rời rạc và sẽ chỉ liên quan đến các quy trình được gắn với vỏ hiện tại:

```
$ ps
  PID TTY          TIME CMD
 2318 pts/0    00:00:00 bash
 2443 pts/0    00:00:00 ps
```

Thông tin được hiển thị sẽ liên quan đến mã định danh quy trình (PID), cửa sổ dòng lệnh mà trong đó quy trình được chạy (TTY), thời gian quy trình sử dụng CPU (TIME) và lệnh bắt đầu quy trình (CMD).

Một khoá chuyển hữu ích cho `ps` là `-f` được dùng để hiển thị danh sách định dạng đầy đủ:

```
$ ps -f
UID          PID  PPID  C  STIME TTY          TIME CMD
carol        2318  1682  0  08:38 pts/1    00:00:00 bash
carol        2443  2318  0  08:46 pts/1    00:00:00 ps -f
```

Khi kết hợp với các khoá chuyển khác, `-f` sẽ hiển thị mối quan hệ giữa các quy trình mẹ và quy trình con:

```
$ ps -uf
```

```

USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
carol    2318  0.0  0.1  21336  5140 pts/1    Ss   08:38   0:00  bash
carol    2492  0.0  0.0  38304  3332 pts/1    R+   08:51   0:00  \_ ps -uf
carol    1780  0.0  0.1  21440  5412 pts/0    Ss   08:28   0:00  bash
carol    2291  0.0  0.7 305352 28736 pts/0    Sl+  08:35   0:00  \_ emacs index.en.adoc
-nw
(...)

```

Tương tự như vậy, `ps` có thể hiển thị phần trăm bộ nhớ được sử dụng khi được gọi với khóa chuyển `-v`:

```

$ ps -v
  PID TTY          STAT TIME  MAJFL  TRS   DRS   RSS %MEM COMMAND
 1163 tty2      Ss1+  0:00    1    67 201224 5576  0.1 /usr/lib/gdm3/gdm-x-session (...)
(...)

```

NOTE

Một lệnh trực quan hấp dẫn khác có thể hiển thị hệ thống phân cấp của các quy trình là `pstree`. Nó có trong tất cả các bản phân phối lớn.

Thông tin Quy trình trong Thư mục `/proc`

Chúng ta đã biết hệ thống tệp `/proc`. `/proc` bao gồm một thư mục con được đánh số cho mọi quy trình đang chạy trong hệ thống (số này chính là PID của quy trình):

```

carol@debian:~# ls /proc
1    108  13  17  21  27  354  41  665  8  9
10   109  14  173  22  28  355  42  7  804  915
103  11  140  18  23  29  356  428  749  810  918
104  111  148  181  24  3  367  432  75  811
105  112  149  19  244  349  370  433  768  83
106  115  15  195  25  350  371  5  797  838
107  12  16  2  26  353  404  507  798  899
(...)

```

Do đó, tất cả các thông tin về một quy trình cụ thể đều sẽ có trong thư mục của nó. Hãy cùng liệt kê nội dung của quy trình đầu tiên — với PID là 1 (đầu ra đã được cắt bớt để dễ đọc hơn):

```

# ls /proc/1/
attr          cmdline          environ  io              mem            ns
autogroup     comm             exe      limits         mountinfo     numa_maps
auxv          coredump_filter fd          loginuid       mounts        oom_adj

```

...

Bạn có thể kiểm tra quá trình thực thi:

```
# cat /proc/1/cmdline; echo
/sbin/init
```

Như bạn có thể thấy, mã nhị phân phân cấp các quy trình là `/sbin/init`.

NOTE

Các lệnh có thể được nối bằng dấu chấm phẩy (;). Mục đích của việc sử dụng lệnh `echo` ở trên là cung cấp một dòng mới. Hãy thử và chạy lệnh `cat /proc/1/cmdline` để thấy được sự khác biệt.

Tải Hệ thống

Mỗi quy trình trên một hệ thống đều có khả năng tiêu tốn tài nguyên hệ thống. Việc tải hệ thống sẽ tổng hợp tải trọng tổng thể của hệ thống thành một chỉ số duy nhất. Bạn có thể xem tải trọng hiện tại bằng lệnh `uptime`:

```
$ uptime
22:12:54 up 13 days, 20:26, 1 user, load average: 2.91, 1.59, 0.39
```

Ba chữ số cuối cùng sẽ cho biết trọng tải trung bình tương ứng của hệ thống trong một phút vừa qua (2,91), năm phút vừa qua (1,59) và mười lăm phút vừa qua (0,39).

Mỗi con số này sẽ cho ta biết có bao nhiêu quy trình đang chờ tài nguyên của CPU hoặc các hoạt động đầu vào/đầu ra hoàn tất. Điều này có nghĩa là các quy trình này đã sẵn sàng để chạy nếu chúng nhận được các tài nguyên tương ứng.

Ghi nhật ký Hệ thống và Thông báo của Hệ thống

Ngay khi hạt nhân và các quy trình bắt đầu thực thi và giao tiếp với nhau, rất nhiều thông tin sẽ được tạo ra. Hầu hết trong số chúng sẽ được gửi đến các tệp nhật ký, hoặc đơn giản hơn là *logs*.

Nếu không ghi nhật ký, việc tìm kiếm một sự kiện xảy ra trên máy chủ sẽ khiến các quản trị viên hệ thống phải đau đầu; do đó, việc có một cách thức chuẩn hóa và tập trung để theo dõi bất kỳ sự kiện nào đã xảy ra trên hệ thống là một việc rất quan trọng. Bên cạnh đó, nhật ký là yếu tố quyết định và hữu ích trong việc xử lý sự cố và bảo mật, cũng như là một nguồn dữ liệu đáng tin cậy để hiểu thống kê của hệ thống và đưa ra dự đoán xu hướng.

Ghi nhật ký bằng nhật ký hệ thống Daemon

Theo truyền thống, các thông báo của hệ thống vẫn sẽ được quản lý bởi cơ sở ghi nhật ký tiêu chuẩn — syslog — hoặc bất kỳ dẫn xuất nào của nó — syslog-ng hoặc rsyslog. Nhật ký Daemon sẽ thu thập thông báo từ các dịch vụ và chương trình khác và sẽ lưu trữ chúng trong tệp nhật ký, thường là `/var/log`. Tuy nhiên, một số dịch vụ sẽ đảm nhận việc ghi nhật ký của riêng chúng (ví dụ như máy chủ web Apache HTTPD). Tương tự như vậy, nhân Linux sử dụng bộ đệm vòng trong bộ nhớ để lưu trữ dữ liệu nhật ký của nó.

Các Tệp Nhật ký trong `/var/log`

Vì nhật ký là dữ liệu sẽ thay đổi theo thời gian nên chúng thường được tìm thấy trong `/var/log`.

Nếu bạn tìm hiểu về `/var/log`, bạn sẽ nhận ra rằng tên của nhật ký — ở một mức độ nào đó — khá là dễ hiểu. Một số ví dụ bao gồm:

`/var/log/auth.log`

Lưu trữ thông tin về việc xác thực (authentication).

`/var/log/kern.log`

Lưu trữ thông tin về hạt nhân.

`/var/log/syslog`

Lưu trữ thông tin hệ thống.

`/var/log/message`

Lưu trữ dữ liệu hệ thống và ứng dụng.

NOTE

Tên và nội dung chính xác của các tệp nhật ký có thể sẽ khác nhau giữa các bản phân phối Linux.

Truy cập Tệp Nhật ký

Khi tìm hiểu về các tệp nhật ký, hãy nhớ rằng bạn phải người dùng gốc (nếu bạn không có quyền đọc) và sử dụng một lệnh nhấn chẳng hạn như `less`;

```
# less /var/log/messages
```

```
Jun  4 18:22:48 debian liblogging-stdlog: [origin software="rsyslogd" swVersion="8.24.0" x-  
pid="285" x-info="http://www.rsyslog.com"] rsyslogd was HUPed
```

```
Jun 29 16:57:10 debian kernel: [  0.000000] Linux version 4.9.0-8-amd64 (debian-  
kernel@lists.debian.org) (gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1) ) #1 SMP  
Debian 4.9.130-2 (2018-10-27)
```

```
Jun 29 16:57:10 debian kernel: [ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-8-amd64 root=/dev/sda1 ro quiet
```

Ngoài ra, bạn có thể sử dụng `tail` với khóa chuyển `-f` để đọc các thông báo gần đây nhất của tệp và tự động hiển thị các dòng mới khi chúng được thêm vào:

```
# tail -f /var/log/messages
Jul  9 18:39:37 debian kernel: [ 2.350572] RAPL PMU: hw unit of domain psys 2^-0 Joules
Jul  9 18:39:37 debian kernel: [ 2.512802] input: VirtualBox USB Tablet as
/devices/pci0000:00/0000:00:06.0/usb1/1-1/1-1:1.0/0003:80EE:0021.0001/input/input7
Jul  9 18:39:37 debian kernel: [ 2.513861] Adding 1046524k swap on /dev/sda5. Priority:-
1 extents:1 across:1046524k FS
Jul  9 18:39:37 debian kernel: [ 2.519301] hid-generic 0003:80EE:0021.0001:
input,hidraw0: USB HID v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-1/input0
Jul  9 18:39:37 debian kernel: [ 2.623947] snd_intel8x0 0000:00:05.0: white list rate for
1028:0177 is 48000
Jul  9 18:39:37 debian kernel: [ 2.914805] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not
ready
Jul  9 18:39:39 debian kernel: [ 4.937283] e1000: enp0s3 NIC Link is Up 1000 Mbps Full
Duplex, Flow Control: RX
Jul  9 18:39:39 debian kernel: [ 4.938493] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link
becomes ready
Jul  9 18:39:40 debian kernel: [ 5.315603] random: crng init done
Jul  9 18:39:40 debian kernel: [ 5.315608] random: 7 urandom warning(s) missed due to
ratelimiting
```

Bạn sẽ thấy đầu ra ở định dạng sau:

- Dấu thời gian
- Tên máy chủ mà tin nhắn đến từ đó
- Tên chương trình/dịch vụ tạo ra thông báo
- PID của chương trình đã tạo thông báo
- Mô tả của hành động đã xảy ra

Hầu hết các tệp nhật ký sẽ được viết bằng văn bản thuần túy; tuy nhiên, một số tệp có thể sẽ chứa các dữ liệu nhị phân như trường hợp của `/var/log/wtmp` — nó lưu trữ dữ liệu liên quan đến các lần đăng nhập thành công. Bạn có thể sử dụng lệnh `file` để xác định cụ thể:

```
$ file /var/log/wtmp
```

```
/var/log/wtmp: dBase III DBT, version number 0, next free block index 8
```

Các tệp này thường được đọc bằng các lệnh đặc biệt. Lệnh `last` được sử dụng để diễn giải dữ liệu trong `/var/log/wtmp`:

```
$ last
carol  tty2      :0          Thu May 30 10:53  still logged in
reboot system boot 4.9.0-9-amd64 Thu May 30 10:52  still running
carol  tty2      :0          Thu May 30 10:47 - crash (00:05)
reboot system boot 4.9.0-9-amd64 Thu May 30 09:11  still running
carol  tty2      :0          Tue May 28 08:28 - 14:11 (05:42)
reboot system boot 4.9.0-9-amd64 Tue May 28 08:27 - 14:11 (05:43)
carol  tty2      :0          Mon May 27 19:40 - 19:52 (00:11)
reboot system boot 4.9.0-9-amd64 Mon May 27 19:38 - 19:52 (00:13)
carol  tty2      :0          Mon May 27 19:35 - down (00:03)
reboot system boot 4.9.0-9-amd64 Mon May 27 19:34 - 19:38 (00:04)
```

NOTE

Tương tự như `/var/log/wtmp`, `/var/log/btmp` lưu trữ thông tin về các lần đăng nhập không thành công và lệnh đặc biệt để đọc nội dung của nó là `lastb`.

Nhật ký Quay vòng

Các tệp nhật ký có thể sẽ tăng lên rất nhiều trong thời gian vài tuần hoặc vài tháng và sẽ chiếm hết dung lượng ổ đĩa trống. Để giải quyết vấn đề này, ta có thể sử dụng tiện ích `logrotate`. Nó sẽ thực hiện luân chuyển hoặc quay vòng nhật ký, tức là di chuyển các tệp nhật ký sang một tên mới, lưu trữ và/hoặc nén chúng, đôi khi là gửi chúng qua email cho quản trị viên hệ thống và cuối cùng là xóa chúng đi khi chúng đã cũ. Các quy ước được sử dụng để đặt tên cho các tệp nhật ký quay vòng này rất đa dạng (ví dụ: thêm hậu tố ngày); tuy nhiên, cách phổ biến hơn là chỉ thêm một hậu tố là một số nguyên:

```
# ls /var/log/apache2/
access.log error.log error.log.1 error.log.2.gz other_vhosts_access.log
```

Hãy lưu ý cách `error.log.2.gz` đã được nén bằng `gzip` (do đó có hậu tố `.gz`).

Bộ đệm vòng của Hạt nhân

Bộ đệm vòng của hạt nhân là một cấu trúc dữ liệu có kích thước cố định để ghi lại các thông báo khởi động hạt nhân cũng như mọi thông báo của hạt nhân đang hoạt động. Chức năng của bộ đệm này — một chức năng rất quan trọng — là ghi nhật ký tất cả các thông báo của hạt nhân được tạo

ra khi khởi động — khi `syslog` chưa khả dụng. Lệnh `dmesg` sẽ in bộ đệm vòng của hạt nhân (trước đây cũng được lưu trữ trong `/var/log/dmesg`). Do phần mở rộng của bộ đệm vòng, lệnh này thường được sử dụng kết hợp với tiện ích lọc văn bản `grep` hoặc lệnh nhấn như `less`. Chẳng hạn, để tìm kiếm thông báo khởi động:

```
$ dmesg | grep boot
[ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro quiet
[ 0.000000] smpboot: Allowing 1 CPUs, 0 hotplug CPUs
[ 0.000000] Kernel command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID=5216e1e4-ae0e-441f-b8f5-8061c0034c74 ro quiet
[ 0.144986] AppArmor: AppArmor disabled by boot time parameter
(...)
```

NOTE

Khi bộ đệm vòng của hạt nhân phát triển cùng với các thông báo mới theo thời gian, những thông báo cũ nhất sẽ mất đi.

Nhật ký hệ thống: `systemd-journald`

Kể từ năm 2015, `systemd` đã thay thế `SysV Init` và trở thành trình quản lý dịch vụ và hệ thống *thực tế* trong hầu hết các bản phân phối Linux chính. Kết quả là trình nhật ký ẩn `journald` đã trở thành thành phần ghi nhật ký tiêu chuẩn, thay thế trình nhật ký hệ thống trong hầu hết các khía cạnh. Dữ liệu không còn được lưu trữ ở dạng văn bản thuần túy mà là ở dạng nhị phân. Do đó, tiện ích `journalctl` sẽ cần thiết trong việc đọc nhật ký. Trên hết, `journald` tương thích với nhật ký hệ thống và có thể được tích hợp với nhật ký hệ thống.

`journalctl` là tiện ích được sử dụng để đọc và truy vấn cơ sở dữ liệu nhật ký của `systemd`. Nếu được gọi mà không có tùy chọn, nó sẽ in toàn bộ nhật ký:

```
# journalctl
-- Logs begin at Tue 2019-06-04 17:49:40 CEST, end at Tue 2019-06-04 18:13:10 CEST. --
jun 04 17:49:40 debian systemd-journald[339]: Runtime journal (/run/log/journal/) is 8.0M,
max 159.6M, 151.6M free.
jun 04 17:49:40 debian kernel: microcode: microcode updated early to revision 0xcc, date =
2019-04-01
Jun 04 17:49:40 debian kernel: Linux version 4.9.0-8-amd64 (debian-kernel@lists.debian.org)
(gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1) )
Jun 04 17:49:40 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-8-amd64
root=/dev/sda1 ro quiet
(...)
```

Tuy nhiên, nếu được gọi với khoá chuyển `-k` hoặc `--dmesg`, nó sẽ tương đương với việc sử dụng lệnh `dmesg`:

```
# journalctl -k
[ 0.000000] Linux version 4.9.0-9-amd64 (debian-kernel@lists.debian.org) (gcc version
6.3.0 20170516 (Debian 6.3.0-18+deb9u1) ) #1 SMP Debian 4.9.168-1+deb9u2 (2019-05-13)
[ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64 root=UUID=5216e1e4-ae0e-
441f-b8f5-8061c0034c74 ro quiet
(...)
```

Các tùy chọn thú vị khác cho `journalctl` bao gồm:

-b, --boot

Hiển thị thông tin khởi động.

-u

Hiển thị thông báo về một đơn vị được chỉ định. Một đơn vị có thể được định nghĩa đại khái là bất kỳ tài nguyên nào được xử lý bởi `systemd`. Ví dụ: `journalctl -u apache2.service` được sử dụng để đọc thông báo về máy chủ web `apache2`.

-f

Hiển thị các thông báo nhật ký gần đây nhất và tiếp tục in các mục mới khi chúng được thêm vào nhật ký—giống như `tail -f`.

Bài tập Hướng dẫn

1. Hãy xem phần liệt kê của lệnh `top` sau đây và trả lời các câu hỏi sau:

```
carol@debian:~$ top

top - 13:39:16 up 31 min,  1 user,  load average: 0.12, 0.15, 0.10
Tasks:  73 total,   2 running,  71 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.1 us,  0.4 sy,   0.0 ni, 98.6 id,   0.0 wa,   0.0 hi,   0.0 si,   0.0 st
KiB Mem : 1020332 total,  698700 free,   170664 used,   150968 buff/cache
KiB Swap: 1046524 total, 1046524 free,     0 used.  710956 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
  605 nobody    20   0 1137620 132424 34256 S   6.3  13.0   1:47.24 ntopng
  444 www-data  20   0  364780   4132  2572 S   0.3   0.4    0:00.44 apache2
  734 root       20   0   95212   7004  6036 S   0.3   0.7    0:00.36 sshd
  887 carol     20   0   46608   3680  3104 R   0.3   0.4    0:00.03 top
    1 root       20   0   56988   6688  5240 S   0.0   0.7    0:00.42 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    0:00.09 ksoftirqd/0
    4 root       20   0     0     0     0 S   0.0   0.0    0:00.87 kworker/0:0
(...)
```

◦ Quy trình nào đã được bắt đầu bởi người dùng `carol`?

◦ Bạn nên truy cập thư mục ảo nào của `/proc` để tìm kiếm dữ liệu liên quan đến lệnh `top`?

◦ Quy trình nào được chạy đầu tiên? Làm cách nào để biết?

◦ Hãy hoàn thành bảng chỉ định các vùng của đầu ra của lệnh `top` có các thông tin sau:

Thông tin về...	Khu Tổng hợp	Khu nhiệm vụ
Bộ nhớ		
Bộ nhớ trao đổi		
PID		
Thời lượng CPU		

Thông tin về...	Khu Tổng hợp	Khu nhiệm vụ
Lệnh		

2. Lệnh nào được sử dụng để đọc nhật ký nhị phân sau đây?

- `/var/log/wtmp`

- `/var/log/btmp`

- `/run/log/journal/2a7d9730cd3142f4b15e20d6be631836/system.journal`

3. Kết hợp với lệnh `grep`, bạn sẽ sử dụng lệnh nào để tìm hiểu thông tin sau về hệ thống Linux của mình?

- Lần cuối cùng khi hệ thống được khởi động lại (`wtmp`)

- Ổ cứng nào được cài đặt (`kern.log`)

- Lần đăng nhập cuối cùng (`auth.log`)

4. Bạn sẽ sử dụng hai lệnh nào để hiển thị bộ đếm vòng của hạt nhân?

5. Hãy cho biết vị trí của các thông điệp nhật ký sau:

- `Jul 10 13:37:39 debian dbus[303]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'`

<code>/var/log/auth.log</code>	
<code>/var/log/kern.log</code>	
<code>/var/log/syslog</code>	
<code>/var/log/messages</code>	

- `Jul 10 11:23:58 debian kernel: [1.923349] usbhid: USB HID core driver`

/var/log/auth.log	
/var/log/kern.log	
/var/log/syslog	
/var/log/messages	

- Jul 10 14:02:53 debian sudo: pam_unix(sudo:session): session opened for user root by carol(uid=0)

/var/log/auth.log	
/var/log/kern.log	
/var/log/syslog	
/var/log/messages	

- Jul 10 11:23:58 debian NetworkManager[322]: <info> [1562750638.8672] NetworkManager (version 1.6.2) is starting...

/var/log/auth.log	
/var/log/kern.log	
/var/log/syslog	
/var/log/messages	

6. journalctl có thông tin truy vấn về các đơn vị sau không?

Đơn vị	Lệnh
ssh	
networking	
rsyslog	
cron	

Bài tập Mở rộng

1. Hãy xem lại đầu ra của lệnh `top` từ các bài tập hướng dẫn và trả lời các câu hỏi sau:

- Bạn sẽ làm theo hai bước nào để ngắt máy chủ web *apache*?

- Trong phần tóm tắt, làm cách nào để có thể hiển thị thông tin về bộ nhớ vật lý và bộ nhớ trao đổi bằng các thanh quy trình?

- Bây giờ, hãy sắp xếp các quy trình theo mức sử dụng bộ nhớ:

- Bây giờ bạn đã có thông tin bộ nhớ được hiển thị trong thanh quy trình và các quy trình được sắp xếp theo mức sử dụng bộ nhớ; hãy lưu các cấu hình này để lấy chúng làm mặc định vào lần tới khi bạn sử dụng `top`:

- Tập nào sẽ lưu cài đặt cấu hình của `top`'s? Nó ở đâu? Làm thế nào để có thể kiểm tra sự tồn tại của nó?

2. Hãy tìm hiểu về lệnh `exec` trong Bash. Hãy chứng minh chức năng của nó bằng cách bắt đầu một phiên Bash, tìm quy trình Bash bằng `ps`, sau đó chạy `exec /bin/sh` và tìm kiếm lại quy trình có cùng PID.

3. Hãy thực hiện theo các bước sau để khám phá các sự kiện của hạt nhân và trình quản lý thiết bị động của `udev`:

- Hãy cắm nóng ổ USB vào máy tính của bạn, chạy `dmesg` và hãy chú ý đến những dòng cuối cùng. Dòng cuối cùng là gì?

- Ghi nhớ đầu ra từ lệnh trước, hãy chạy `ls /dev/sd*` và đảm bảo rằng ổ USB của bạn xuất hiện trong danh sách. Đầu ra sẽ là gì?

- Bây giờ, hãy tháo ổ USB và chạy lại `dmesg`. Dòng cuối cùng sẽ là gì?

- Hãy chạy lại `ls /dev/sd*` và đảm bảo rằng thiết bị của bạn đã biến mất khỏi danh sách.

Đầu ra sẽ là gì?

Tóm tắt

Trong phạm vi của việc lưu trữ dữ liệu, các chủ đề sau đã được thảo luận trong bài học này: quản lý quy trình, ghi nhật ký và thông báo của hệ thống.

Về quản lý quy trình, chúng ta đã học được về:

- Các chương trình sẽ tạo ra các quy trình và các quy trình tồn tại trong một hệ thống phân cấp.
- Mỗi quy trình đều có một mã định danh duy nhất (PID) và một mã định danh quy trình gốc (PPID).
- `top` là một lệnh rất hữu ích để khám phá các quy trình đang chạy của hệ thống một cách linh hoạt và mang tính tương tác.
- `ps` có thể được sử dụng để lấy hình ảnh tức thời của các quy trình đang chạy hiện tại trong hệ thống.
- Thư mục `/proc` bao gồm các thư mục cho mọi quy trình đang chạy trong hệ thống và được đặt tên theo các PID của chúng.
- Khái niệm về trung bình tải trọng hệ thống rất hữu ích trong việc kiểm tra việc sử dụng/ làm quá tải CPU.

Về tác vụ ghi nhật ký hệ thống, chúng ta phải nhớ rằng:

- Nhật ký là một tệp ghi lại các sự kiện trong hệ thống. Nhật ký vô cùng quan trọng trong việc khắc phục sự cố.
- Theo truyền thống, việc ghi nhật ký sẽ được xử lý bởi các dịch vụ đặc biệt như `syslog`, `syslog-ng` hoặc `rsyslog`. Tuy nhiên, một số chương trình lại sử dụng nhật ký ẩn của riêng chúng.
- Vì nhật ký là dữ liệu dạng biến nên chúng được lưu trong `/var` và — đôi khi — tên của chúng có thể sẽ gợi ý cho ta về nội dung của chúng (`kern.log`, `auth.log`, v.v.)
- Hầu hết các nhật ký đều được viết bằng văn bản thuần túy và có thể được đọc bằng bất kỳ trình soạn thảo văn bản nào, miễn là bạn có quyền phù hợp. Tuy nhiên, một vài trong số chúng là dữ liệu nhị phân và phải được đọc bằng các lệnh đặc biệt.
- Để tránh các vấn đề về dung lượng ổ đĩa, việc *ghi nhật ký quay vòng* có thể được thực hiện bởi tiện ích `logrotate`.
- Đối với hạt nhân, nó sử dụng cấu trúc dữ liệu xoay vòng — bộ đệm vòng — để lưu giữ các thông báo khởi động (các thông báo cũ sẽ mất dần theo thời gian).
- Trình quản lý hệ thống và dịch vụ `systemd` đã thay thế System V init trong hầu như tất cả các bản phân phối bằng `journald` và trở thành dịch vụ ghi nhật ký tiêu chuẩn.

- Để đọc nhật ký của systemd, ta cần có tiện ích `journalctl`.

Các lệnh được dùng trong bài này:

cat

Nối/in nội dung tệp.

dmesg

In bộ đệm vòng của hạt nhân.

echo

Hiển thị một dòng văn bản hoặc một dòng mới.

file

Xác định loại tệp.

grep

In các dòng khớp với một mẫu.

last

Hiển thị danh sách người dùng đăng nhập lần cuối.

less

Hiển thị nội dung của tệp từng trang một.

ls

Liệt kê nội dung thư mục.

journalctl

Truy vấn nhật ký systemd.

tail

Hiển thị các dòng cuối cùng của một tệp.

Đáp án Bài tập Hướng dẫn

1. Hãy xem phần liệt kê của `top` sau đây và trả lời các câu hỏi sau:

```
carol@debian:~$ top

top - 13:39:16 up 31 min,  1 user,  load average: 0.12, 0.15, 0.10
Tasks:  73 total,   2 running,  71 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.1 us,  0.4 sy,  0.0 ni, 98.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 1020332 total,  698700 free,  170664 used,  150968 buff/cache
KiB Swap: 1046524 total, 1046524 free,    0 used.  710956 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
  605 nobody    20   0 1137620 132424 34256 S   6.3  13.0   1:47.24 ntopng
  444 www-data  20   0  364780   4132  2572 S   0.3   0.4   0:00.44 apache2
  734 root       20   0   95212   7004  6036 S   0.3   0.7   0:00.36 sshd
  887 carol     20   0   46608   3680  3104 R   0.3   0.4   0:00.03 top
    1 root       20   0   56988   6688  5240 S   0.0   0.7   0:00.42 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   0:00.09 ksoftirqd/0
    4 root       20   0     0     0     0 S   0.0   0.0   0:00.87 kworker/0:0
(...)
```

- Các quy trình nào đã được bắt đầu bởi người dùng `carol`?

Trả lời: Chỉ có một: `top`.

- Bạn nên truy cập thư mục ảo nào của `/proc` để tìm kiếm dữ liệu liên quan đến lệnh `top`?

Trả lời: `/proc/887`

- Quy trình nào được chạy đầu tiên? Làm cách nào để biết?

Trả lời: `systemd`. Bởi vì nó là quy trình có PID #1.

- Hãy hoàn thành bảng chỉ định các vùng của đầu ra của lệnh `top` có các thông tin sau:

Thông tin về...	Khu Tổng hợp	Khu nhiệm vụ
Memory	Có	Có
Swap	Có	Không
PID	Không	Có

Thông tin về...	Khu Tổng hợp	Khu nhiệm vụ
CPU time	Có	Có
Commands	Không	Có

2. Lệnh nào được sử dụng để đọc nhật ký nhị phân sau đây?

- `/var/log/wtmp`

Trả lời: `last`

- `/var/log/btmp`

Trả lời: `lastb`

- `/run/log/journal/2a7d9730cd3142f4b15e20d6be631836/system.journal`

Trả lời: `journalctl`

3. Kết hợp với `grep`, bạn sẽ sử dụng lệnh nào để tìm hiểu thông tin sau về hệ thống Linux của mình?

- Lần cuối cùng khi hệ thống được khởi động lại (`wtmp`)

Trả lời: `last`

- Ổ cứng nào được cài đặt (`kern.log`)

Trả lời: `less /var/log/kern.log`

- Lần đăng nhập cuối cùng (`auth.log`)

Trả lời: `less /var/log/auth.log`

4. Bạn sẽ sử dụng hai lệnh nào để hiển thị bộ đệm vòng của hạt nhân?

`dmesg` và `journalctl -k` (còn có cả `journalctl --dmesg`).

5. Hãy cho biết vị trí của các thông điệp nhật ký sau:

- `Jul 10 13:37:39 debian dbus[303]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'`

`/var/log/auth.log`

`/var/log/kern.log`

/var/log/syslog	X
/var/log/messages	

- Jul 10 11:23:58 debian kernel: [1.923349] usbhid: USB HID core driver

/var/log/auth.log	
/var/log/kern.log	X
/var/log/syslog	
/var/log/messages	X

- Jul 10 14:02:53 debian sudo: pam_unix(sudo:session): session opened for user root by carol(uid=0)

/var/log/auth.log	X
/var/log/kern.log	
/var/log/syslog	
/var/log/messages	

- Jul 10 11:23:58 debian NetworkManager[322]: <info> [1562750638.8672] NetworkManager (version 1.6.2) is starting...

/var/log/auth.log	
/var/log/kern.log	
/var/log/syslog	
/var/log/messages	X

6. journalctl có thông tin truy vấn về các đơn vị sau không?

Đơn vị	Lệnh
ssh	journalctl -u ssh.service
networking	journalctl -u networking.service
rsyslog	journalctl -u rsyslog.service
cron	journalctl -u cron.service

Đáp án Bài tập Mở rộng

1. Hãy xem lại đầu ra của lệnh `top` từ các bài tập hướng dẫn và trả lời các câu hỏi sau:

- Bạn sẽ làm theo hai bước nào để ngắt máy chủ web *apache*?

Đầu tiên, hãy nhấn `k`; sau đó hãy thêm giá trị `kill`.

- Trong phần tóm tắt, làm cách nào để có thể hiển thị thông tin về bộ nhớ vật lý và bộ nhớ trao đổi bằng các thanh quy trình?

Bằng cách nhấn `m` một hoặc hai lần.

- Bây giờ, hãy sắp xếp các quy trình theo mức sử dụng bộ nhớ:

`M`

- Bây giờ bạn đã có thông tin bộ nhớ được hiển thị trong thanh quy trình và các quy trình được sắp xếp theo mức sử dụng bộ nhớ; hãy lưu các cấu hình này để lấy chúng làm mặc định vào lần tới khi bạn sử dụng `top`:

`W`

- Tập nào sẽ lưu cài đặt cấu hình của `top`'s? Nó ở đâu? Làm thế nào để có thể kiểm tra sự tồn tại của nó?

Tập này là `~/ .config/procps/toprc` và nằm trong thư mục chính của người dùng (`~`). Vì nó là một tập ẩn (nó nằm trong một thư mục có tên bắt đầu bằng dấu chấm) nên chúng ta có thể kiểm tra sự tồn tại của nó bằng `ls -a` (liệt kê tất cả các tập). Tập này có thể được tạo bằng cách nhấn `Shift + W` khi đang ở trong `top`.

2. Hãy tìm hiểu về lệnh `exec` trong Bash. Hãy chứng minh chức năng của nó bằng cách bắt đầu một phiên Bash, tìm quy trình Bash bằng `ps`, sau đó chạy `exec /bin/sh` và tìm kiếm lại quy trình có cùng PID.

`exec` sẽ thay thế một quy trình bằng một lệnh khác. Trong ví dụ sau, chúng ta có thể thấy rằng quy trình Bash được thay thế bằng `/bin/sh` (thay vì `/bin/sh` trở thành quy trình `con`):

```
$ echo $$
19877
$ ps auxf | grep 19877 | head -1
carol 19877 0.0 0.0 7448 3984 pts/25 Ss 21:17 0:00 \_ bash
$ exec /bin/sh
```



```
sh-5.0$ ps auxf | grep 19877 | head -1
carol 19877 0.0 0.0 7448 3896 pts/25 Ss 21:17 0:00 \_ /bin/sh
```

3. Hãy thực hiện theo các bước sau để khám phá các sự kiện của hạt nhân và trình quản lý thiết bị động của udev:

- Hãy cắm nóng ổ USB vào máy tính của bạn, chạy `dmesg` và hãy chú ý đến những dòng cuối cùng. Dòng cuối cùng là gì?

Bạn sẽ nhận được một dòng có dạng như `[1967.700468] sd 6:0:0:0: [sdb] Attached SCSI diable disk.`

- Ghi nhớ đầu ra từ lệnh trước, hãy chạy `ls /dev/sd*` và đảm bảo rằng ổ USB của bạn xuất hiện trong danh sách. Đầu ra sẽ là gì?

Tùy thuộc vào số lượng thiết bị được kết nối với hệ thống của bạn, bạn sẽ nhận được một dòng có dạng như `/dev/sda /dev/sda1 /dev/sdb /dev/sdb1 /dev/sdb2`. Trong trường hợp này, chúng ta sẽ tìm thấy ổ USB (`/dev/sdb`) và hai phân vùng của nó (`/dev/sdb1` và `/dev/sdb2`).

- Bây giờ, hãy tháo ổ USB và chạy lại `dmesg`. Dòng cuối cùng sẽ là gì?

Bạn sẽ nhận được một dòng có dạng như `[2458.881695] usb 1-9: USB disconnect, device number 6.`

- Hãy chạy lại `ls /dev/sd*` và đảm bảo rằng thiết bị của bạn đã biến mất khỏi danh sách. Đầu ra sẽ là gì?

Trong trường hợp này: `/dev/sda /dev/sda1`.



**Linux
Professional
Institute**

4.4 Máy tính của bạn trên mạng

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 4.4](#)

Khối lượng

2

Các lĩnh vực kiến thức chính

- Internet, mạng, bộ định tuyến
- Truy vấn cấu hình máy khách DNS
- Truy vấn cấu hình mạng

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

- `route`, `ip route show`
- `ifconfig`, `ip addr show`
- `netstat`, `ss`
- `/etc/resolv.conf`, `/etc/hosts`
- IPv4, IPv6
- `ping`
- `host`



4.4 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	4 Hệ điều hành Linux
Mục tiêu:	4.4 Máy tính trên Mạng
Bài:	1 trên 1

Giới thiệu

Trong thế giới ngày nay, bất kỳ loại thiết bị máy tính nào cũng sẽ thực hiện việc trao đổi thông tin qua mạng. Trọng tâm của khái niệm mạng máy tính là các kết nối vật lý giữa một hoặc nhiều thiết bị đồng cấp với nó. Các kết nối này được gọi là *liên kết*; chúng là kết nối cơ bản nhất giữa hai thiết bị khác nhau. Các liên kết có thể được thiết lập thông qua các phương tiện khác nhau, chẳng hạn như cáp đồng, sợi quang, sóng vô tuyến hoặc tia laze.

Mỗi liên kết sẽ được kết nối với một giao diện của thiết bị. Mỗi thiết bị có thể có nhiều giao diện và do đó được kết nối với nhiều liên kết. Thông qua các liên kết này, các máy tính có thể tạo thành một mạng - một cộng đồng nhỏ nơi các thiết bị có thể kết nối trực tiếp với nhau. Có rất nhiều ví dụ về các mạng như vậy trên thế giới. Để có thể giao tiếp ngoài phạm vi của mạng tầng liên kết, các thiết bị phải sử dụng bộ định tuyến. Hãy coi các mạng tầng liên kết như các hòn đảo được kết nối bởi các bộ định tuyến kết nối — giống như những cây cầu mà thông tin sẽ di chuyển qua để có thể đến được một hòn đảo hẻo lánh khác - tức một thiết bị khác.

Mô hình này dẫn đến một số tầng mạng khác nhau:

Tầng liên kết

Xử lý giao tiếp giữa các thiết bị được kết nối trực tiếp.

Tầng mạng

Xử lý định tuyến bên ngoài các mạng riêng lẻ và địa chỉ duy nhất của các thiết bị bên ngoài mạng tầng liên kết đơn.

Lớp ứng dụng

Cho phép các chương trình riêng lẻ kết nối với nhau.

Khi mới được phát minh, các mạng máy tính đã sử dụng các phương thức liên lạc chuyển mạch giống như điện thoại. Việc này có nghĩa là một liên kết chuyên dụng và trực tiếp phải được hình thành giữa hai nút để chúng có thể giao tiếp. Phương pháp này đã hoạt động khá tốt; tuy nhiên, nó lại yêu cầu tất cả không gian trên một liên kết chỉ để hai máy chủ giao tiếp.

Sau cùng, các mạng máy tính đã chuyển sang một phương thức khác gọi là *chuyển mạch gói*. Ở phương thức này, dữ liệu được nhóm lại với một tiêu đề chứa thông tin về nơi nó xuất phát và đích đến của nó. Thông tin nội dung thực tế sẽ được chứa trong khung này và sẽ được gửi qua liên kết đến người nhận được chỉ định trong tiêu đề của khung. Điều này cho phép nhiều thiết bị chia sẻ một liên kết và giao tiếp gần như cùng một lúc.

Mạng Tầng Liên kết

Công việc của bất kỳ một gói thông tin nào cũng là đưa thông tin từ nguồn đến đích thông qua một liên kết kết nối cả hai thiết bị với nhau. Các thiết bị này cần một phương thức để nhận dạng nhau. Đây chính là mục đích của *địa chỉ tầng liên kết*. Trong mạng Ethernet, *Địa chỉ kiểm soát truy cập phương tiện* (MAC) được sử dụng để xác định các thiết bị riêng lẻ. Một địa chỉ MAC sẽ bao gồm 48 bit. Chúng không phải là duy nhất trong phạm vi toàn cầu và không thể được sử dụng để xác định các địa chỉ ngang hàng ngoài liên kết hiện tại. Do đó, các địa chỉ này không thể được sử dụng để định tuyến các gói tin đến các liên kết khác. Bên nhận gói sẽ kiểm tra xem địa chỉ đích có khớp với tầng liên kết của chính nó hay không và nếu khớp, bên nhận sẽ xử lý gói. Nếu không thì gói tin sẽ bị loại bỏ. Ngoại lệ đối với quy tắc này là *gói tin quảng bá* (một gói tin được gửi tới mọi máy trong một mạng cục bộ nhất định); gói tin này sẽ luôn được chấp nhận.

Lệnh `ip link show` sẽ hiển thị danh sách tất cả các giao diện mạng khả dụng và địa chỉ tầng liên kết của chúng cũng như một số thông tin khác về kích thước gói tối đa:

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

```
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT
group default qlen 1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
```

Đầu ra ở trên cho thấy thiết bị có hai giao diện, `lo` và `ens33`. `lo` là thiết bị vòng lặp và có địa chỉ MAC là `00:00:00:00:00:00`, trong khi `ens33` là giao diện ethernet và có địa chỉ MAC là `00:0c:29:33:3b:25`.

Mạng IPv4

Để truy cập các trang web như Google hoặc Twitter, để kiểm tra email hoặc cho phép các doanh nghiệp kết nối với nhau, các gói cần có khả năng chuyển vùng từ mạng tầng liên kết này sang mạng tầng liên kết khác. Thông thường, các mạng này chỉ được kết nối gián tiếp với một số mạng tầng liên kết trung gian mà các gói phải đi qua để đến được đích đến thực tế của chúng.

Địa chỉ tầng liên kết của giao diện mạng không thể được sử dụng bên ngoài mạng tầng liên kết đó. Vì địa chỉ này vô nghĩa đối với các thiết bị trong các mạng tầng liên kết khác nên phải có một dạng địa chỉ duy nhất trên toàn cầu để thực hiện định tuyến. Cơ chế địa chỉ này cùng với khái niệm tổng thể về định tuyến được triển khai bởi *Giao thức Internet* (IP - Internet Protocol).

NOTE

Một *giao thức* là một tập hợp các quy trình thực hiện một tác vụ sao cho tất cả các bên tuân theo giao thức đều sẽ tương thích với nhau. Một giao thức có thể được coi là định nghĩa của một tiêu chuẩn. Trong điện toán, Giao thức Internet là một tiêu chuẩn được thống nhất để các thiết bị khác nhau được sản xuất bởi các nhà sản xuất khác nhau đều có thể giao tiếp với nhau.

Địa chỉ IPv4

Giống như địa chỉ MAC, địa chỉ IP là một cách để chỉ ra nơi xuất phát và đích đến của một gói dữ liệu. IPv4 là giao thức nguyên sơ. Địa chỉ IPv4 có độ rộng 32 bit và cho số lượng địa chỉ tối đa theo lý thuyết là 4.294.967.296. Tuy nhiên, số lượng địa chỉ mà các thiết bị có thể sử dụng nhỏ hơn nhiều vì một số dải được dành riêng cho các trường hợp sử dụng đặc biệt như địa chỉ quảng bá (được sử dụng để tiếp cận tất cả các máy tham gia một mạng cụ thể), địa chỉ phát đa hướng (tương tự như địa chỉ quảng bá, nhưng mỗi thiết bị sẽ phải bắt sóng như radio) hoặc những thiết bị dành riêng cho mục đích sử dụng cá nhân.

Ở định dạng con người có thể đọc được, các địa chỉ IPv4 có ký hiệu là bốn chữ số được phân tách bằng dấu chấm. Mỗi chữ số có thể nằm trong khoảng từ 0 đến 255. Ví dụ như địa chỉ IP sau:

```
192.168.0.1
```

Về mặt kỹ thuật, mỗi chữ số này đại diện cho tám bit riêng lẻ. Do đó, địa chỉ này cũng có thể được viết như sau:

```
11000000.10101000.00000000.00000001
```

Trong thực tế, ký hiệu thập phân đã được sử dụng. Tuy nhiên, việc biểu diễn bằng bit vẫn là rất quan trọng để có thể hiểu mạng con.

Mạng con IPv4

Để hỗ trợ việc định tuyến, địa chỉ IP có thể được chia thành hai phần: phần mạng và phần máy chủ. Phần mạng sẽ xác định mạng nơi thiết bị đang kết nối và được sử dụng để định tuyến các gói đến mạng đó. Phần máy chủ được sử dụng để xác định cụ thể một thiết bị nhất định trên mạng và chuyển gói tin đến bên nhận cụ thể sau khi nó đã đến được mạng tầng liên kết của nó.

Địa chỉ IP có thể được chia thành các phần mạng con và máy chủ tại bất kỳ thời điểm nào. Cái gọi là *mặt nạ mạng con* sẽ xác định nơi xảy ra sự phân chia này. Hãy xem xét lại biểu diễn nhị phân của địa chỉ IP từ ví dụ trước:

```
11000000.10101000.00000000.00000001
```

Bây giờ, đối với địa chỉ IP này, mặt nạ mạng con sẽ đặt từng bit thuộc phần mạng thành 1 và mỗi bit thuộc phần máy chủ thành 0:

```
11111111.11111111.11111111.00000000
```

Trong thực tế, mặt nạ mạng sẽ được viết bằng ký hiệu thập phân:

```
255.255.255.0
```

Điều này có nghĩa là mạng này nằm trong khoảng từ 192.168.0.0 đến 192.168.0.255. Hãy lưu ý rằng ba số đầu tiên có tất cả các bit được đặt trong mặt nạ mạng vẫn không thay đổi trong các địa chỉ IP.

Cuối cùng, có một ký hiệu thay thế cho mặt nạ mạng con được gọi là *Định tuyến Liên miền không phân loại* (CIDR - Classless Inter-Domain Routing). Ký hiệu này sẽ chỉ cho biết có bao nhiêu bit được đặt trong mặt nạ mạng con và thêm số này vào địa chỉ IP. Trong ví dụ này, 24 trong số 32 bit được đặt thành 1 trong mặt nạ mạng con. Điều này có thể được thể hiện bằng ký hiệu CIDR là

192.168.0.1/24

Địa chỉ IPv4 riêng tư

Như đã đề cập trước đó, một số phần của không gian địa chỉ IPv4 sẽ được dành riêng cho các trường hợp sử dụng đặc biệt. Một trong những trường hợp sử dụng này là gán địa chỉ riêng tư. Các mạng con sau được dành riêng cho địa chỉ riêng tư:

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

Bất kỳ ai cũng có thể sử dụng địa chỉ của các mạng con này. Tuy nhiên, các mạng con này không thể được định tuyến trên internet công cộng vì chúng có khả năng được sử dụng bởi nhiều mạng cùng một lúc.

Ngày nay, hầu hết các mạng đều sử dụng các địa chỉ nội bộ này. Chúng cho phép giao tiếp nội bộ mà không cần gán địa chỉ bên ngoài. Hầu hết các kết nối internet ngày nay chỉ đi kèm với một địa chỉ IPv4 bên ngoài duy nhất. Bộ định tuyến sẽ ánh tất cả các địa chỉ bên trong tới địa chỉ IP bên ngoài duy nhất đó khi chuyển tiếp các gói tới mạng internet. Điều này được gọi là *Dịch địa chỉ mạng* (NAT - Network Address Translation). Trường hợp đặc biệt của NAT mà trong đó bộ định tuyến ánh xạ các địa chỉ bên trong tới một địa chỉ IP bên ngoài duy nhất đôi khi được gọi là *giả trang*. Điều này cho phép mọi thiết bị trong mạng thiết lập các kết nối mới với bất kỳ địa chỉ IP toàn cầu nào trên mạng internet.

NOTE

Trong trường hợp giả trang, các thiết bị nội bộ sẽ không thể được tham chiếu từ internet vì chúng không có địa chỉ hợp lệ trên toàn cầu. Tuy nhiên, đây không phải là một tính năng bảo mật. Ngay cả khi giả trang thì ta vẫn phải có tường lửa.

Cấu hình địa chỉ IPv4

Có hai cách chính để định cấu hình địa chỉ IPv4 trên máy tính. Một là chỉ định địa chỉ theo cách thủ công, hai là bằng cách sử dụng *Giao thức Cấu hình Máy chủ động* (DHCP - Dynamic Host Configuration Protocol) để cấu hình tự động.

Khi sử dụng DHCP, một máy chủ trung tâm sẽ kiểm soát việc phân bổ địa chỉ cho các thiết bị. Máy chủ cũng có thể cung cấp cho các thiết bị các thông tin khác về mạng, chẳng hạn như địa chỉ IP của máy chủ DNS, địa chỉ IP của bộ định tuyến mặc định hoặc để khởi động hệ điều hành từ mạng trong trường hợp thiết lập phức tạp. DHCP sẽ được bật theo mặc định trên nhiều hệ thống; do đó, bạn có thể đã có sẵn địa chỉ IP khi kết nối với mạng.

Địa chỉ IP cũng có thể được thêm thủ công vào giao diện bằng cách sử dụng lệnh `ip addr add`. Ở đây, chúng ta sẽ thêm địa chỉ `192.168.0.5` vào giao diện `ens33`. Mạng sử dụng mặt nạ mạng `255.255.255.0`, tương đương với `/24` trong ký hiệu CIDR:

```
$ sudo ip addr add 192.168.0.5/255.255.255.0 dev ens33
```

Bây giờ chúng ta có thể xác minh rằng địa chỉ đã được thêm bằng cách sử dụng lệnh `ip addr show`:

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
25: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.5/24 192.168.0.255 scope global ens33
        valid_lft forever preferred_lft forever
    inet6 fe80::010c:29ff:fe33:3b25/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Đầu ra ở trên đã cho thấy cả giao diện `lo` và giao diện `ens33` cùng với địa chỉ của nó được gán bằng lệnh ở trên.

Để xác minh khả năng tiếp cận của thiết bị, ta có thể sử dụng lệnh `ping`. Lệnh sẽ gửi một loại tin nhắn đặc biệt được gọi là *yêu cầu phản hồi* mà trong đó, người gửi yêu cầu người nhận phải phản hồi. Nếu kết nối giữa hai thiết bị có thể được thiết lập thành công, người nhận sẽ gửi lại phản hồi xác nhận kết nối đã được thiết lập:

```
$ ping -c 3 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=2.16 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=1.85 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=3.41 ms

--- 192.168.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 5ms
```



```
rtt min/avg/max/mdev = 1.849/2.473/3.410/0.674 ms
```

Tham số `-c 3` sẽ khiến ping dừng lại sau khi gửi ba yêu cầu phản hồi. Nếu không, ping sẽ tiếp tục chạy mãi và phải dừng lại bằng cách nhấn `Ctrl + C`.

Định tuyến IPv4

Định tuyến là một quá trình mà trong đó một gói sẽ đi từ mạng nguồn đến mạng đích. Mỗi thiết bị sẽ duy trì một bảng định tuyến chứa thông tin về việc mạng IP nào có thể được truy cập trực tiếp thông qua phần đính kèm của thiết bị với các mạng tầng liên kết, và mạng IP nào có thể được truy cập bằng cách chuyển các gói tới bộ định tuyến. Cuối cùng, *định tuyến mặc định* sẽ xác định một bộ định tuyến có thể nhận tất cả các gói không khớp với bất kỳ định tuyến nào khác.

Khi thiết lập kết nối, thiết bị sẽ tra cứu địa chỉ IP của mục tiêu trong bảng định tuyến của nó. Nếu tìm thấy một mục phù hợp với địa chỉ, gói sẽ được gửi đến mạng tầng liên kết tương ứng hoặc được chuyển đến bộ định tuyến được chỉ định trong bảng định tuyến.

Bản thân các bộ định tuyến cũng duy trì các bảng định tuyến của riêng mình. Khi nhận được một gói, một bộ định tuyến cũng tra cứu địa chỉ đích trong bảng định tuyến của chính nó và gửi gói đến bộ định tuyến tiếp theo. Điều này sẽ được lặp lại cho đến khi gói đến được bộ định tuyến trên mạng đích. Mỗi bộ định tuyến tham gia vào hành trình này được gọi là *hop*. Bộ định tuyến cuối cùng này sẽ tìm thấy một liên kết được kết nối trực tiếp cho địa chỉ đích trong bảng định tuyến của nó và gửi các gói đến giao diện đích.

Hầu hết các mạng gia đình chỉ có một lối thoát, đó là bộ định tuyến duy nhất đến từ *nhà cung cấp dịch vụ internet* (ISP - Internet Service Provider). Trong trường hợp này, một thiết bị chỉ chuyển tiếp trực tiếp tất cả các gói không dành cho mạng nội bộ tới bộ định tuyến gia đình, sau đó bộ định tuyến này sẽ gửi các gói tới bộ định tuyến của nhà cung cấp để chuyển tiếp thêm. Đây là một ví dụ về định tuyến mặc định.

Lệnh `ip route show` sẽ liệt kê bảng định tuyến IPv4 hiện tại:

```
$ ip route show
127.0.0.0/8 via 127.0.0.1 dev lo0
192.168.0.0/24 dev ens33 scope link
```

Để thêm một định tuyến mặc định, tất cả những gì ta cần là địa chỉ nội bộ của bộ định tuyến sẽ là cổng mặc định. Ví dụ: nếu bộ định tuyến có địa chỉ là `192.168.0.1` thì lệnh sau sẽ thiết lập nó làm định tuyến mặc định:

```
$ sudo ip route add default via 192.168.0.1
```

Để xác minh, hãy chạy lại `ip route show`:

```
$ ip route show
default via 192.168.0.1 dev ens33
127.0.0.0/8 via 127.0.0.1 dev lo0
192.168.0.0/24 dev ens33 scope link
```

Mạng IPv6

IPv6 được thiết kế để giải quyết những hạn chế của IPv4, chủ yếu là về việc thiếu hụt địa chỉ khi ngày càng có nhiều thiết bị được đưa lên mạng. Tuy nhiên, IPv6 cũng bao gồm các tính năng khác, ví dụ như các giao thức mới để cấu hình mạng tự động. Thay vì 32 bit cho mỗi địa chỉ, IPv6 sử dụng 128 bit. Điều này cho phép nó cung cấp khoảng 2^{128} địa chỉ. Tuy nhiên, giống như IPv4, số lượng địa chỉ có thể sử dụng duy nhất trên toàn cầu nhỏ hơn rất nhiều so với con số trên do các phần phân bổ được dành riêng cho các mục đích sử dụng khác. Số lượng địa chỉ khổng lồ này ám chỉ việc nó có thừa địa chỉ công cộng cho mọi thiết bị hiện đang được kết nối với internet và thậm chí là nhiều hơn thế nữa, do đó giảm đi nhu cầu giả trang và các vấn đề liên quan như chậm trễ trong quá trình dịch và khả năng không thể kết nối trực tiếp được với các thiết bị giả trang.

Địa chỉ IPv6

Khi được viết ra, các địa chỉ sẽ sử dụng 8 nhóm gồm 4 chữ số thập lục phân, mỗi nhóm được phân tách bằng một dấu hai chấm:

```
2001:0db8:0000:abcd:0000:0000:0000:7334
```

NOTE

Các chữ số thập lục phân nằm trong khoảng từ 0 đến f, vì vậy mỗi chữ số có thể chứa một trong 16 giá trị khác nhau.

Để dễ hiểu hơn, các số 0 đứng đầu trong mỗi nhóm có thể được bỏ đi khi viết ra; tuy nhiên, mỗi nhóm vẫn phải chứa ít nhất một chữ số:

```
2001:db8:0:abcd:0:0:0:7334
```

Trong trường hợp nhiều nhóm chỉ chứa các chữ số 0 nối đuôi nhau, chúng có thể được thay thế hoàn toàn bằng '::':

```
2001:db8:0:abcd::7334
```

Tuy nhiên, thay thế này chỉ có thể xảy ra một lần trong mỗi một địa chỉ.

Tiền tố IPv6

64 bit đầu tiên của địa chỉ IPv6 được gọi là *tiền tố định tuyến*. Tiền tố sẽ được các bộ định tuyến sử dụng để xác định thiết bị thuộc về mạng nào và từ đó xác định dữ liệu cần được gửi qua đường dẫn nào. Việc chia mạng con luôn xảy ra bên trong tiền tố. Các ISP thường cung cấp cho khách hàng của họ các tiền tố /48 hoặc /58, để lại cho họ 16 hoặc 8 bit dành cho mạng con nội bộ.

Có ba loại tiền tố chính trong IPv6:

Địa chỉ toàn cầu duy nhất

Trong đó tiền tố được gán từ các khối dành riêng cho địa chỉ trên toàn cầu. Những địa chỉ này sẽ hợp lệ trên toàn bộ không gian internet.

Địa chỉ cục bộ duy nhất

Có thể không được định tuyến trong internet. Tuy nhiên, chúng có thể được định tuyến nội bộ trong một tổ chức. Các địa chỉ này được sử dụng trong một mạng để đảm bảo các thiết bị vẫn có địa chỉ ngay cả khi không có kết nối internet. Chúng tương đương với các dải địa chỉ riêng tư từ IPv4. 8 bit đầu tiên luôn là `fc` hoặc `fd`, tiếp theo là 40 bit được tạo ngẫu nhiên.

Địa chỉ Liên kết Địa phương

Chỉ hợp lệ trên một liên kết cụ thể. Mỗi giao diện mạng hỗ trợ IPv6 đều có một địa chỉ như vậy, bắt đầu bằng `fe80`. Các địa chỉ này được IPv6 sử dụng nội bộ để yêu cầu các địa chỉ bổ sung bằng cách sử dụng cấu hình tự động và để tìm các máy tính khác trong cùng mạng bằng giao thức Tìm Hàng xóm (Neighbor Discovery).

Định danh Giao diện IPv6

Trong khi tiền tố xác định thiết bị nằm trong mạng nào, mã định danh giao diện được sử dụng để liệt kê các thiết bị trong một mạng. 64 bit cuối cùng trong địa chỉ IPv6 sẽ tạo thành mã định danh giao diện, giống hệt như các bit cuối cùng của địa chỉ IPv4.

Khi địa chỉ IPv6 được gán thủ công, mã định danh giao diện sẽ được đặt thành một phần của địa chỉ. Khi sử dụng cấu hình địa chỉ tự động, mã định danh giao diện có thể được chọn ngẫu nhiên hoặc được lấy từ địa chỉ tầng liên kết của thiết bị. Điều này khiến một biến thể của địa chỉ tầng liên kết xuất hiện trong địa chỉ IPv6.

Cấu hình địa chỉ IPv6

Giống như IPv4, địa chỉ IPv6 có thể được gán thủ công hoặc tự động. Tuy nhiên, IPv6 có hai loại cấu hình tự động khác nhau, DHCPv6 và *Cấu hình Tự động Địa chỉ phi Trạng thái* (SLAAC - Stateless Address Autoconfiguration).

SLAAC là phương pháp dễ dàng hơn trong hai phương pháp tự động và được tích hợp ngay vào tiêu chuẩn IPv6. Các tin nhắn sử dụng *Giao thức Tìm Hàng xóm* mới cho phép các thiết bị tìm thấy nhau và yêu cầu các thông tin liên quan đến mạng. Các thông tin này sẽ được gửi bởi các bộ định tuyến và chúng có thể sẽ chứa các tiền tố IPv6 mà các thiết bị có thể sử dụng bằng cách kết hợp chúng với một mã định danh giao diện mà chúng chọn, miễn là địa chỉ kết quả chưa được sử dụng. Các thiết bị sẽ không cung cấp phản hồi cho bộ định tuyến về các địa chỉ thực đã tạo ra.

Mặt khác, DHCPv6 là phiên bản đã được cập nhật của DHCP để phù hợp hơn với những thay đổi của IPv6. Nó khiến việc kiểm soát thông tin được cung cấp cho khách hàng trở nên hiệu quả hơn, chẳng hạn như cho phép cùng một địa chỉ được trao cho cùng một khách hàng mỗi lần và gửi nhiều tùy chọn hơn cho khách hàng so với SLAAC. Với DHCPv6, khách hàng cần có sự đồng ý của máy chủ DHCP để có thể sử dụng địa chỉ.

Phương pháp gán thủ công địa chỉ IPv6 cho giao diện cũng sẽ giống như với IPv4:

```
$ sudo ip addr add 2001:db8:0:abcd:0:0:0:7334/64 dev ens33
```

Để xác minh việc gán đã thành công, lệnh `ip addr show` tương tự sẽ được sử dụng:

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
25: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
   link/ether 00:0c:29:33:3b:25 brd ff:ff:ff:ff:ff:ff
   inet 192.168.0.5/24 192.168.0.255 scope global ens33
       valid_lft forever preferred_lft forever
   inet6 fe80::010c:29ff:fe33:3b25/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
   inet6 2001:db8:0:abcd::7334/64 scope global
       valid_lft forever preferred_lft forever
```

Ở đây, chúng ta cũng có thể thấy địa chỉ liên kết cục bộ là `fe80::010c:29ff:fe33:3b25/64`.

Giống như IPv4, lệnh `ping` cũng có thể được sử dụng để xác nhận khả năng kết nối của các thiết bị thông qua IPv6:

```
$ ping 2001:db8:0:abcd::1
PING 2001:db8:0:abcd::1(2001:db8:0:abcd::1) 56 data bytes
64 bytes from 2001:db8:0:abcd::1: icmp_seq=1 ttl=64 time=0.030 ms
64 bytes from 2001:db8:0:abcd::1: icmp_seq=2 ttl=64 time=0.040 ms
64 bytes from 2001:db8:0:abcd::1: icmp_seq=3 ttl=64 time=0.072 ms

--- 2001:db8:0:abcd::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 43ms
rtt min/avg/max/mdev = 0.030/0.047/0.072/0.018 ms
```

NOTE

Trên một số hệ thống Linux, `ping` không hỗ trợ IPv6. Thay vào đó, các hệ thống này cung cấp một lệnh chuyên dụng là `ping6`.

Để xác minh lại địa chỉ liên kết cục bộ, hãy sử dụng lại `ping`. Nhưng vì tất cả các giao diện đều sử dụng tiền tố `fe80::/64`, giao diện chính xác phải được chỉ định cùng với địa chỉ:

```
$ ping6 -c 1 fe80::010c:29ff:fe33:3b25%ens33
PING fe80::010c:29ff:fe33:3b25(fe80::010c:29ff:fe33:3b25) 56 data bytes
64 bytes from fe80::010c:29ff:fe33:3b25%ens33: icmp_seq=1 ttl=64 time=0.049 ms

--- fe80::010c:29ff:fe33:3b25 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.049/0.049/0.049/0.000 ms
```

DNS

Địa chỉ IP rất khó nhớ và nghe không được hấp dẫn lắm nếu bạn đang muốn tiếp thị một dịch vụ hoặc một sản phẩm. Đây là lúc *Hệ Thống Tên Miền* (DNS - Domain Name System) phát huy tác dụng. Ở dạng đơn giản nhất, DNS là một danh bạ điện thoại được phân phối đánh dấu các tên miền thân thiện dễ nhớ (chẳng hạn như `example.com`) tới các địa chỉ IP. Ví dụ: khi người dùng điều hướng đến một trang web, họ sẽ nhập tên máy chủ DNS như một phần của URL. Sau đó, trình duyệt web sẽ gửi tên DNS tới bất kỳ trình phân giải DNS nào đã được định cấu hình. Trình phân giải DNS đó sẽ lần lượt tìm ra địa chỉ tương quan với tên miền. Sau đó, trình phân giải sẽ trả lời bằng địa chỉ đó và trình duyệt web sẽ cố gắng truy cập máy chủ web tại địa chỉ IP đó.

Trình phân giải mà Linux sử dụng để tra cứu dữ liệu DNS được định cấu hình trong tệp cấu hình

/etc/resolv.conf:

```
$ cat /etc/resolv.conf
search lpi
nameserver 192.168.0.1
```

Khi trình phân giải thực hiện tra cứu tên, trước tiên, nó sẽ kiểm tra tệp /etc/hosts để xem liệu nó có chứa địa chỉ cho tên được yêu cầu hay không. Nếu có, nó sẽ trả về địa chỉ đó và không liên hệ với DNS. Điều này cho phép các quản trị viên mạng cung cấp độ phân giải tên mà không cần phải mất công cấu hình một máy chủ DNS hoàn chỉnh. Mỗi dòng trong tệp đó sẽ chứa một địa chỉ IP, theo sau là một hoặc nhiều tên:

```
127.0.0.1      localhost.localdomain  localhost
::1           localhost.localdomain  localhost
192.168.0.10  server
2001:db8:0:abcd::f server
```

Để thực hiện tra cứu trong DNS, hãy sử dụng lệnh `host`:

```
$ host learning.lpi.org
learning.lpi.org has address 208.94.166.198
```

Thông tin chi tiết hơn có thể được truy xuất bằng lệnh `dig`:

```
$ dig learning.lpi.org
; <<>> DiG 9.14.3 <<>> learning.lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21525
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 2ac55879b1adef30a93013705d3306d2128571347df8eadf (bad)
;; QUESTION SECTION:
;learning.lpi.org.      IN  A

;; ANSWER SECTION:
learning.lpi.org.      550 IN  A    208.94.166.198
```

```
;; Query time: 3 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Sat Jul 20 14:20:21 EST 2019
;; MSG SIZE rcvd: 89
```

Ở đây, chúng ta cũng đã thấy tên của các loại bản ghi DNS, trong trường hợp này là A cho IPv4.

Ổ nối

Ổ nối là điểm giao tiếp cuối để hai chương trình tương tác với nhau. Nếu ổ nối được kết nối qua mạng, các chương trình có thể chạy trên các thiết bị khác nhau, chẳng hạn như trình duyệt web chạy trên máy tính xách tay của người dùng và máy chủ web chạy trong trung tâm dữ liệu của công ty.

Có ba loại ổ nối chính:

Ổ nối Unix

Kết nối các quy trình đang chạy trên cùng một thiết bị.

Ổ nối UDP (User Datagram Protocol - Giao thức Dữ liệu Người dùng)

Kết nối các ứng dụng bằng giao thức, nhanh nhưng không linh hoạt.

Ổ nối TCP (Transmission Control Protocol - Giao thức Điều khiển Truyền nhận)

Đáng tin cậy hơn ổ cắm UDP, chẳng hạn như trong việc xác nhận việc nhận dữ liệu.

Ổ nối Unix chỉ có thể kết nối các ứng dụng chạy trên cùng một thiết bị. Tuy nhiên, ổ nối TCP và UDP có thể kết nối qua mạng. TCP cho phép luồng dữ liệu luôn đến theo thứ tự chính xác như khi nó được gửi. UDP thì chỉ biết gửi; gói sẽ được gửi đi nhưng việc có đến được đúng địa chỉ hay không lại không thể đảm bảo. Tuy nhiên, UDP không có công suất bổ sung như TCP; điều này khiến nó trở thành một sự lựa chọn hoàn hảo cho các ứng dụng có độ trễ thấp như trò chơi điện tử trực tuyến.

Cả TCP và UDP đều sử dụng các cổng để giải quyết nhiều ổ nối trên cùng một địa chỉ IP. Mặc dù cổng nguồn cho mỗi một kết nối thường là ngẫu nhiên, nhưng các cổng đích phải được chuẩn hóa cho mỗi dịch vụ cụ thể. Ví dụ: HTTP thường được lưu trữ ở cổng 80, HTTPS chạy trên cổng 443. SSH, một giao thức để đăng nhập an toàn vào hệ thống Linux từ xa, "lắng nghe" ở cổng 22.

Lệnh `ss` cho phép quản trị viên điều tra tất cả các ổ nối trên máy tính Linux. Nó sẽ cho ta thấy mọi thứ từ địa chỉ nguồn, địa chỉ đích và loại. Một trong những tính năng tốt nhất của nó là sử dụng các bộ lọc để người dùng có thể giám sát các ổ nối ở bất kỳ trạng thái kết nối nào họ muốn. `ss` có thể được chạy với một tập hợp các tùy chọn cũng như biểu thức bộ lọc để giới hạn thông tin được

hiển thị.

Khi được thực thi mà không có bất kỳ tùy chọn nào, lệnh sẽ hiển thị danh sách của tất cả các ổ nối đã thiết lập. Việc sử dụng tùy chọn `-p` sẽ cho ra thông tin về quá trình (process) sử dụng từng ổ nối. Tùy chọn `-s` hiển thị tóm tắt (summary) các ổ nối. Có nhiều tùy chọn khác có sẵn cho công cụ này, nhưng tập hợp các tùy chọn chính cuối cùng là `-4` và `-6` để thu hẹp giao thức IP thành IPv4 hoặc IPv6 tương ứng, `-t` và `-u` cho phép quản trị viên chọn ổ nối TCP hoặc UDP và `-l` để chỉ hiển thị ổ cắm "lắng nghe" (listen) các kết nối mới.

Ví dụ, lệnh sau sẽ liệt kê tất cả các ổ nối TCP hiện đang được sử dụng:

```
$ ss -t
State      Recv-Q  Send-Q   Local Address:Port    Peer Address:Port
ESTAB      0        0        192.168.0.5:49412     192.168.0.1:https
ESTAB      0        0        192.168.0.5:37616     192.168.0.1:https
ESTAB      0        0        192.168.0.5:40114     192.168.0.1:https
ESTAB      0        0        192.168.0.5:54948     192.168.0.1:imap
...
```


Bài tập Hướng dẫn

1. Một kỹ sư mạng được yêu cầu gán hai địa chỉ IP cho giao diện `ens33` của máy chủ, một địa chỉ IPv4 (`192.168.10.10/24`) và một địa chỉ IPv6 (`2001:0:0:abcd:0:8a2e:0370:7334/64`). Họ sẽ phải nhập những lệnh nào để đạt được điều này?

2. Địa chỉ nào trong các lựa chọn dưới đây là địa chỉ riêng tư?

192.168.10.1	
120.56.78.35	
172.16.57.47	
10.100.49.162	
200.120.42.6	

3. Bạn sẽ thêm gì vào tệp máy chủ để gán `192.168.0.15` cho `example.com`?

4. Lệnh sau có tác dụng gì?

```
sudo ip -6 route add default via 2001:db8:0:abcd::1
```

Bài tập Mở rộng

1. Hãy đặt tên cho loại bản ghi DNS được sử dụng để phục vụ các yêu cầu sau:

- Dữ liệu văn bản

- Tra cứu địa chỉ IP ngược

- Miền không có địa chỉ riêng và phải dựa vào miền khác

- Máy chủ Thư điện tử

2. Linux có một tính năng gọi là bắc cầu, nó làm gì và có tác dụng như thế nào?

3. Tùy chọn nào cần được cung cấp cho lệnh `ss` để xem tất cả các ổ nối UDP đã thiết lập?

4. Lệnh nào hiển thị tóm tắt của tất cả các ổ nối đang chạy trên thiết bị Linux?

5. Đầu ra sau đây được tạo bởi lệnh từ bài tập trước. Có bao nhiêu ổ nối TCP và UDP đang hoạt động?

```
Total: 978 (kernel 0)
TCP: 4 (estab 0, closed 0, orphaned 0, synrecv 0, timewait 0/0), ports 0

Transport Total      IP          IPv6
*      0          -          -
RAW    1           0           1
UDP    7           5           2
TCP    4           3           1
INET   12          8           4
FRAG   0           0           0
```

Tóm tắt

Bài học này đã nói về kết nối mạng máy tính Linux của bạn. Đầu tiên, chúng ta đã tìm hiểu về các cấp độ khác nhau của mạng: * Tầng liên kết kết nối trực tiếp các thiết bị. * Tầng mạng cung cấp định tuyến giữa các mạng và không gian địa chỉ toàn cầu. * Tầng ứng dụng nơi các ứng dụng kết nối với nhau. Chúng ta đã thấy cách IPv4 và IPv6 được sử dụng để định địa chỉ cho các máy tính riêng lẻ và cách TCP và UDP liệt kê các ổ nối được các ứng dụng sử dụng để kết nối với nhau. Chúng ta cũng đã học cách sử dụng DNS để phân giải tên thành địa chỉ IP.

Các lệnh được dùng trong bài tập:

dig

Truy vấn thông tin DNS và cung cấp thông tin chi tiết về các truy vấn và phản hồi của DNS.

host

Truy vấn thông tin DNS và cung cấp đầu ra rút gọn.

ip

Định cấu hình mạng trên Linux, bao gồm giao diện mạng, địa chỉ và định tuyến.

ping

Kiểm tra kết nối với một thiết bị từ xa.

ss

Hiển thị thông tin liên quan đến ổ nối.

Đáp án Bài tập Hướng dẫn

1. Một kỹ sư mạng được yêu cầu gán hai địa chỉ IP cho giao diện `ens33` của máy chủ, một địa chỉ IPv4 (`192.168.10.10/24`) và một địa chỉ IPv6 (`2001:0:0:abcd:0:8a2e:0370:7334/64`). Họ sẽ phải nhập những lệnh nào để đạt được điều này?

```
sudo ip addr add 192.168.10.10/24 dev ens33
sudo ip addr add 2001:0:0:abcd:0:8a2e:0370:7334/64 dev ens33
```

2. Địa chỉ nào trong các lựa chọn dưới đây là địa chỉ riêng tư?

192.168.10.1	X
120.56.78.35	
172.16.57.47	X
10.100.49.162	X
200.120.42.6	

3. Bạn sẽ thêm gì vào tệp máy chủ để gán `192.168.0.15` cho `example.com`?

```
192.168.0.15 example.com
```

4. Lệnh sau có tác dụng gì?

```
sudo ip -6 route add default via 2001:db8:0:abcd::1
```

Nó sẽ thêm một định tuyến mặc định vào bảng để gửi tất cả lưu lượng IPv6 tới bộ định tuyến với địa chỉ nội bộ là `2001:db8:0:abcd::1`.

Đáp án Bài tập Mở rộng

1. Hãy đặt tên cho loại bản ghi DNS được sử dụng để phục vụ các yêu cầu sau:

- Dữ liệu văn bản

TXT

- Tra cứu địa chỉ IP ngược

PTR

- Miền không có địa chỉ riêng và phải dựa vào miền khác

CNAME

- Máy chủ Thư điện tử

MX

2. Linux có một tính năng gọi là bắc cầu, nó làm gì và có tác dụng như thế nào?

Một cầu nối sẽ kết nối nhiều giao diện mạng. Tất cả các giao diện kết nối vào một cầu nối có thể giao tiếp như thể chúng đang kết nối vào cùng một mạng tầng liên kết: Tất cả các thiết bị sẽ sử dụng địa chỉ IP từ cùng một mạng con và không cần phải có bộ định tuyến để kết nối với nhau.

3. Tùy chọn nào cần được cung cấp cho lệnh `ss` để xem tất cả các ổ nối UDP đã thiết lập?

Tùy chọn `-u` sẽ hiển thị tất cả các ổ nối UDP đã thiết lập.

4. Lệnh nào hiển thị tóm tắt của tất cả các ổ nối đang chạy trên thiết bị Linux?

Lệnh `ss -s` hiển thị tóm tắt tất cả các ổ nối.

5. Đầu ra sau đây được tạo bởi lệnh từ bài tập trước. Có bao nhiêu ổ nối TCP và UDP đang hoạt động?

```
Total: 978 (kernel 0)
TCP: 4 (estab 0, closed 0, orphaned 0, synrecv 0, timewait 0/0), ports 0

Transport Total      IP          IPV6
*      0          -           -
RAW    1          0           1
UDP    7          5           2
```

TCP	4	3	1	
INET	12	8	4	
FRAG	0	0	0	

11 ổ nối TCP và UDP đang hoạt động.



Chủ đề 5: Quyền bảo mật và Tệp



Linux
Professional
Institute

5.1 Bảo mật Cơ bản và Xác định Loại Người dùng

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 5.1](#)

Khối lượng

2

Các lĩnh vực kiến thức chính

- Người dùng gốc và Người dùng tiêu chuẩn
- Người dùng hệ thống

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

- `/etc/passwd`, `/etc/shadow`, `/etc/group`
- `id`, `last`, `who`, `w`
- `sudo`, `su`



5.1 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	5 Bảo mật và Quyền trong Tập
Mục tiêu:	5.1 Bảo mật cơ bản và Xác định kiểu Người dùng
Bài:	1 trên 1

Giới thiệu

Bài học này sẽ tập trung vào thuật ngữ cơ bản về tài khoản, tính năng kiểm soát truy cập và cơ chế bảo mật của các hệ thống Linux cục bộ, các công cụ giao diện dòng lệnh (CLI) trên hệ thống Linux dùng trong kiểm soát truy cập bảo mật cơ bản và các tệp cơ bản để hỗ trợ tài khoản người dùng và nhóm, bao gồm cả các công cụ dùng để nâng cao đặc quyền cơ bản.

Bảo mật cơ bản trong các hệ thống Linux được mô phỏng theo các biện pháp kiểm soát truy cập của Unix; các biện pháp này mặc dù đã gần 50 năm tuổi nhưng vẫn khá hiệu quả so với một số hệ điều hành tiêu dùng phổ biến thuộc các dòng mới hơn hẳn. Ngay cả một số hệ điều hành dựa trên Unix phổ biến khác cũng có xu hướng “tự do thay đổi” để tập trung vào tính “dễ truy cập”, nhưng Linux thì lại không như vậy.

Các giao diện và môi trường máy tính để bàn Linux hiện đại đã đơn giản hóa việc tạo và quản lý người dùng và thường tự động hóa việc chỉ định tính năng kiểm soát truy cập khi người dùng đăng nhập — ví dụ: vào trình hiển thị, âm thanh và các dịch vụ khác — và hầu như không yêu cầu sự can thiệp thủ công của quản trị viên hệ thống. Tuy nhiên, điều quan trọng là ta phải hiểu được các khái niệm cơ bản của một hệ điều hành Linux cơ bản.

Tài khoản

Việc bảo mật liên quan đến nhiều khái niệm; một trong những khái niệm phổ biến nhất là khái niệm chung về tính năng kiểm soát truy cập. Trước khi có thể xử lý các khía cạnh của tính năng kiểm soát truy cập tệp như quyền sở hữu và cấp phép, ta phải hiểu về các khái niệm cơ bản về *tài khoản* người dùng Linux vốn được chia thành nhiều loại.

Mỗi người dùng trên hệ thống Linux đều có một tài khoản riêng; bên cạnh thông tin đăng nhập (như tên người dùng và mật khẩu), tài khoản này cũng xác định cách thức và vị trí mà người dùng có thể tương tác với hệ thống. Đặc quyền và tính năng kiểm soát truy cập sẽ xác định “ranh giới” mà mỗi người dùng có thể hoạt động trong phạm vi đó.

Mã định danh (UID/GID)

Mã định danh Người dùng và Nhóm (UID/GID) là các tham chiếu cơ bản được kê khai cho các tài khoản. Các triển khai ban đầu là các số nguyên 16 bit bị giới hạn (giá trị từ 0 đến 65535), nhưng các hệ thống của thế kỷ 21 đã hỗ trợ các UID và GID 64 bit. Người dùng và nhóm sẽ được kê khai độc lập; do đó, cùng một ID có thể đại diện cho cả người dùng và nhóm.

Mỗi người dùng không chỉ có một UID mà còn có một *GID chính*. GID chính của một người dùng có thể là duy nhất cho riêng người dùng đó và có thể sẽ không được sử dụng bởi bất kỳ người dùng nào khác. Tuy nhiên, nó cũng có thể là một nhóm được chia sẻ bởi nhiều người dùng. Ngoài các nhóm chính này, mỗi người dùng cũng có thể là thành viên của các nhóm khác nữa.

Theo mặc định trên các hệ thống Linux, mọi người dùng sẽ được chỉ định vào một nhóm có cùng tên với tên người dùng và có cùng GID với UID của họ. Ví dụ: ta sẽ tạo một người dùng mới có tên `newuser` và theo mặc định, nhóm mặc định của nó cũng sẽ là `newuser`.

Tài khoản Siêu Người dùng

Trên Linux, tài khoản siêu người dùng là `root` (gốc); tài khoản này luôn có UID 0. Tài khoản siêu người dùng đôi khi còn được gọi là *quản trị viên hệ thống* và sẽ có quyền truy cập và kiểm soát không giới hạn đối với hệ thống, bao gồm cả những người dùng khác.

Nhóm mặc định cho siêu người dùng có GID 0 và cũng được đặt tên là `root`. Thư mục chính dành cho siêu người dùng là một thư mục chuyên dụng cấp cao nhất (tức `/root`) chỉ có thể được truy cập bởi người dùng `root`.

Tài khoản Người dùng Tiêu chuẩn

Tất cả các tài khoản không phải `root` đều là tài khoản người dùng thông thường về mặt kỹ thuật;

nhưng trên hệ thống Linux, thuật ngữ thông dụng *tài khoản người dùng* thường có nghĩa là tài khoản người dùng “thông thường” (không mang đặc quyền). Chúng thường có các thuộc tính sau với một số ngoại lệ chọn lọc:

- UID bắt đầu từ 1000 (4 chữ số), mặc dù một số hệ thống cũ sẽ có thể bắt đầu từ 500.
- Một thư mục chính được chỉ định, thường là thư mục con của `/home`, tùy thuộc vào cấu hình cục bộ.
- một vỏ đăng nhập được chỉ định. Trong Linux, vỏ mặc định thường là *Vỏ Bourne Again* (`/bin/bash`), mặc dù cũng có thể có các vỏ khác nữa.

Nếu tài khoản người dùng không có vỏ hợp lệ trong thuộc tính của chúng, người dùng sẽ không thể mở được vỏ tương tác. Thông thường `/sbin/nologin` được sử dụng như một vỏ không hợp lệ. Điều này có thể mang một chủ ý nhất định nếu người dùng chỉ được xác thực cho các dịch vụ không phải là quyền truy cập bảng điều khiển hoặc SSH, ví dụ như chỉ có quyền truy cập Giao thức truyền tệp SSH (`sftp`).

NOTE

Để tránh nhầm lẫn, thuật ngữ *tài khoản người dùng* sẽ chỉ áp dụng cho tài khoản người dùng tiêu chuẩn hoặc thông thường từ nay về sau. Ví dụ: *tài khoản hệ thống* sẽ được sử dụng để giải thích về một tài khoản người dùng Linux thuộc loại tài khoản người dùng hệ thống.

Tài khoản Hệ thống

Tài khoản hệ thống thường được tạo từ trước khi cài đặt hệ thống. Đây là những cơ sở, chương trình và dịch vụ sẽ không chạy trên tư cách siêu người dùng. Trong một thế giới lý tưởng, tất cả chúng sẽ là tiện ích của hệ điều hành.

Các tài khoản hệ thống không giống nhau nhưng các thuộc tính của chúng thường đều bao gồm:

- UID thường dưới 100 (2 số) hoặc từ 500-1000 (3 số).
- Hoặc là *không* có Thư mục chính chuyên dụng, hoặc là có một thư mục thường không nằm trong `/home`.
- Không có vỏ đăng nhập hợp lệ (thường là `/sbin/nologin`), hiếm có trường hợp ngoại lệ.

Hầu hết các tài khoản hệ thống trên Linux sẽ không bao giờ đăng nhập và không cần vỏ chỉ định trong thuộc tính của chúng. Nhiều quy trình do tài khoản hệ thống sở hữu và thực thi sẽ được trình quản lý hệ thống phân nhánh vào môi trường riêng của chúng, chạy với tài khoản hệ thống được chỉ định. Những tài khoản này thường có các đặc quyền hạn chế, hoặc đa số là *không* có.

NOTE

Theo LPI Linux Essentials, tài khoản hệ thống là UID <1000, với UID 2 hoặc 3 chữ số

(và GID).

Nói chung, các tài khoản hệ thống *không* nên có vỏ đăng nhập hợp lệ. Nếu có, nó sẽ trở thành một vấn đề bảo mật trong hầu hết các trường hợp.

Tài khoản Dịch vụ

Tài khoản Dịch vụ thường được tạo khi dịch vụ được cài đặt và định cấu hình. Tương tự như tài khoản hệ thống, chúng dành cho các tiện ích, chương trình và dịch vụ sẽ không chạy dưới dạng siêu người dùng.

Trong nhiều tài liệu, tài khoản hệ thống và dịch vụ sẽ tương tự như nhau và thường được hoán đổi cho nhau. Điều này bao gồm cả vị trí của các thư mục chính thường nằm ngoài `/home` nếu được chỉ định (các tài khoản dịch vụ thường có nhiều khả năng có thư mục này hơn so với các tài khoản hệ thống) và không có vỏ đăng nhập hợp lệ. Mặc dù không có một định nghĩa cụ thể nào nhưng sự khác biệt chính về cơ bản giữa tài khoản hệ thống và dịch vụ là UID/GID.

Tài khoản hệ thống

UID/GID <100 (2 số) hoặc <500-1000 (3 số)

Tài khoản dịch vụ

UID/GID >1000 (4+ chữ số), nhưng không phải là tài khoản người dùng “tiêu chuẩn” hoặc “thông thường”, một tài khoản dành cho dịch vụ, có UID/GID >1000 (4+ chữ số)

Một số bản phân phối Linux vẫn có các tài khoản dịch vụ đã đặt sẵn với UID <100 và những tài khoản đó cũng có thể được coi là tài khoản hệ thống mặc dù chúng không được tạo khi cài đặt hệ thống. Ví dụ: trên các bản phân phối Linux dựa trên Fedora (bao gồm cả Red Hat), người dùng cho máy chủ Web Apache có UID (và GID) 48; rõ ràng đó là một tài khoản hệ thống mặc dù nó có thư mục chính (thường tại `/usr/share/httpd` hoặc `/var/www/html/`).

NOTE

Theo LPI Linux Essentials, tài khoản hệ thống là UID <1000 và tài khoản người dùng thông thường là UID >1000. Vì tài khoản người dùng thông thường >1000 nên những UID này cũng có thể bao gồm cả các tài khoản dịch vụ.

Vỏ Đăng nhập và Thư mục chính

Một số tài khoản có vỏ đăng nhập trong khi những tài khoản khác không có vì mục đích bảo mật do chúng không yêu cầu quyền truy cập tương tác. Vỏ đăng nhập mặc định trên hầu hết các bản phân phối Linux là *Vỏ Bourne Again* hoặc `bash`; cũng có thể có các vỏ khác như vỏ C (`csh`), vỏ Korn (`ksh`) hoặc vỏ Z (`zsh`) — trên đây chỉ là một vài ví dụ.

Người dùng có thể thay đổi vỏ đăng nhập của mình bằng cách sử dụng lệnh `chsh`. Theo mặc định,

lệnh sẽ chạy ở chế độ tương tác và hiển thị lời nhắc để hỏi vỏ nào sẽ được sử dụng. Câu trả lời phải là một đường dẫn đầy đủ đến tệp nhị phân vỏ như bên dưới:

```
$ chsh
```

```
Changing the login shell for emma
Enter the new value, or press ENTER for the default
Login Shell [/bin/bash]: /usr/bin/zsh
```

Ta cũng có thể chạy lệnh ở chế độ không tương tác với tham số `-s` và theo sau là đường dẫn đến tệp nhị phân như sau:

```
$ chsh -s /usr/bin/zsh
```

Hầu hết các tài khoản đều có một thư mục chính xác định. Trên Linux, đây thường là vị trí duy nhất mà tài khoản người dùng đó có quyền ghi được đảm bảo với một số ngoại lệ (ví dụ: các khu vực hệ thống tệp tạm thời). Tuy nhiên, một số tài khoản được thiết lập có chủ đích để không có bất kỳ quyền ghi nào vào ngay cả thư mục chính của chúng vì mục đích bảo mật.

Nhận Thông tin về Người dùng của bạn

Liệt kê thông tin người dùng cơ bản là một tác vụ phổ biến hàng ngày trên hệ thống Linux. Trong một số trường hợp, người dùng sẽ cần chuyển đổi người dùng và tăng đặc quyền để hoàn thành các tác vụ đặc quyền.

Ngay cả người dùng cũng có khả năng liệt kê các thuộc tính và điểm truy cập từ dòng lệnh bằng cách sử dụng các lệnh bên dưới. Thông tin cơ bản trong một bối cảnh hạn chế không phải là một thao tác đặc quyền.

Việc liệt kê thông tin hiện tại của người dùng tại dòng lệnh chỉ đơn giản là một lệnh gồm hai chữ cái: `id`. Đầu ra sẽ thay đổi tùy theo ID đăng nhập của bạn:

```
$ id
```

```
uid=1024(emma) gid=1024(emma) 1024(emma),20(games),groups=10240(netusers),20480(netadmin)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Trước đó, người dùng (`emma`) có các mã định danh có thể được phân tích như sau:

- `1024` = ID người dùng (UID), theo sau là tên người dùng (tên thông thường hay còn gọi là tên đăng nhập) trong ngoặc đơn.

- 1024 = ID nhóm *chính* (GID), theo sau là tên nhóm (tên thông thường) trong ngoặc đơn.
- Danh sách các GID bổ sung (tên nhóm) mà người dùng cũng có thể có.

Việc liệt kê lần cuối người dùng đăng nhập vào hệ thống được thực hiện bằng lệnh `last`:

```
$ last
emma pts/3      ::1           Fri Jun 14 04:28  still logged in
reboot system boot  5.0.17-300.fc30. Fri Jun 14 04:03  still running
reboot system boot  5.0.17-300.fc30. Wed Jun  5 14:32 - 15:19  (00:46)
reboot system boot  5.0.17-300.fc30. Sat May 25 18:27 - 19:11  (00:43)
reboot system boot  5.0.16-100.fc28. Sat May 25 16:44 - 17:06  (00:21)
reboot system boot  5.0.9-100.fc28.x Sun May 12 14:32 - 14:46  (00:14)
root  tty2          Fri May 10 21:55 - 21:55  (00:00)
...
```

Thông tin được liệt kê trong các cột có thể khác nhau, nhưng một số mục đáng chú ý trong phần liệt kê ở trên là:

- Một người dùng (emma) đã đăng nhập qua mạng (TTY pts/3 giả) và vẫn đang đăng nhập.
- Thời điểm khởi động hiện tại được liệt kê cùng với nhân. Trong ví dụ trên, khoảng 25 phút trước khi người dùng đăng nhập.
- Siêu người dùng (root) đã đăng nhập qua bảng điều khiển ảo (TTY tty2), trong khoảng thời gian ngắn, vào giữa tháng Năm.

Một biến thể của lệnh `last` là lệnh `lastb`, liệt kê tất cả các lần đăng nhập không hợp lệ gần đây nhất.

Các lệnh `who` và `w` chỉ liệt kê các thông tin đăng nhập đang hoạt động trên hệ thống:

```
$ who
emma pts/3      2019-06-14 04:28 (::1)

$ w
 05:43:41 up 1:40, 1 user, load average: 0.25, 0.53, 0.51
USER      TTY      LOGIN@  IDLE   JCPU   PCPU   WHAT
emma pts/3      04:28    1:14m  0.04s  0.04s  -bash
```

Cả hai lệnh đều liệt kê một số thông tin giống nhau. Ví dụ: một người dùng (emma) đã đăng nhập bằng thiết bị TTY giả (pts/3) và thời gian đăng nhập là 04:28.

Lệnh `w` liệt kê nhiều thông tin hơn, bao gồm:

- Thời gian hiện tại và hệ thống đã hoạt động được bao lâu
- Có bao nhiêu người dùng được kết nối
- *Trung bình Tải lượng* trong 1, 5 và 15 phút qua

Và thông tin bổ sung cho mỗi phiên người dùng đang hoạt động.

- Chọn, tổng thời gian sử dụng CPU (IDLE, JCPU và PCPU)
- Quy trình hiện tại (`-bash`). Tổng thời gian sử dụng CPU của quy trình đó là mục cuối cùng (PCPU).

Cả hai lệnh đều có các tùy chọn khác để liệt kê các thông tin bổ sung khác nhau.

Chuyển đổi Người dùng và Nâng cao Đặc quyền

Trong một thế giới lý tưởng, người dùng sẽ không bao giờ cần phải nâng đặc quyền để hoàn thành nhiệm vụ của mình. Hệ thống sẽ luôn “tự hoạt động” và mọi thứ sẽ được định cấu hình cho các loại truy cập khác nhau.

Thật may mắn cho chúng ta bởi Linux — theo mặc định — đã hoạt động như vậy đối với hầu hết những người dùng không phải là quản trị viên hệ thống mặc dù nó luôn tuân theo mô hình bảo mật *ít đặc quyền nhất*.

Tuy nhiên, có những lệnh cho phép nâng đặc quyền khi cần thiết. Hai trong số những lệnh quan trọng nhất là `su` và `sudo`.

Trên hầu hết các hệ thống Linux ngày nay, lệnh `su` chỉ được sử dụng để nâng cấp đặc quyền lên quyền gốc - tức người dùng mặc định - nếu tên người dùng không được chỉ định sau tên lệnh. Mặc dù nó có thể được sử dụng để chuyển sang người dùng khác nhưng đây không phải là cách tốt: người dùng nên đăng nhập từ một hệ thống khác qua mạng, bảng điều khiển vật lý hoặc cửa sổ dòng lệnh trên hệ thống.

```
emma ~$ su -  
Password:  
root ~#
```

Sau khi nhập mật khẩu siêu người dùng (`root`), người dùng sẽ có vỏ siêu người dùng (chú ý dấu `#` ở cuối dấu nhắc lệnh) và sẽ là siêu người dùng (`root`) trên mọi mặt.

Việc chia sẻ mật khẩu là một hành vi bảo mật rất tồi; vì vậy, lệnh `su` sẽ rất ít khi (thậm chí là không bao giờ) cần tới trong một hệ thống Linux hiện đại.

Ký hiệu đô la (\$) sẽ chấm dứt dấu nhắc lệnh cho vỏ người dùng không có đặc quyền, trong khi ký hiệu (#) sẽ kết thúc dấu nhắc lệnh cho dấu nhắc vỏ siêu người dùng (root). Bạn *không bao giờ* nên thay đổi ký tự cuối cùng của bất kỳ dấu nhắc lệnh nào so với tiêu chuẩn “phổ biến toàn cầu” này vì danh pháp này đã được sử dụng trong hầu hết các tài liệu học tập, bao gồm cả những tài liệu này.

WARNING

Đừng bao giờ chuyển sang tài khoản siêu người dùng (root) mà không chuyển tham số hệ vỏ đăng nhập (-). Trừ khi có hướng dẫn rõ ràng khác của Hệ điều hành hoặc nhà cung cấp phần mềm khi `su` được yêu cầu, hãy luôn thực thi `su -` với các ngoại lệ cực kỳ hạn chế. Môi trường người dùng có thể gây ra các sự cố và thay đổi cấu hình không mong muốn khi được sử dụng ở chế độ đầy đủ đặc quyền với tư cách là siêu người dùng.

Vấn đề lớn nhất khi sử dụng `su` để chuyển sang siêu người dùng (root) là gì? Nếu phiên của người dùng thông thường bị xâm phạm, mật khẩu siêu người dùng (root) có thể sẽ bị lấy. Đó là lúc “Switch User Do” (hoặc “Superuser Do”) xuất hiện:

```
$ cat /sys/devices/virtual/dmi/id/board_serial
cat: /sys/devices/virtual/dmi/id/board_serial: Permission denied

$ sudo cat /sys/devices/virtual/dmi/id/board_serial
[sudo] password for emma:
/6789ABC/
```

Trong phần liệt kê trên, người dùng đang cố tra cứu số sê-ri bo mạch hệ thống của họ. Tuy nhiên, quyền truy cập đã bị từ chối vì thông tin đó được đánh dấu là đặc quyền.

Tuy nhiên, bằng cách sử dụng `sudo`, người dùng sẽ nhập mật khẩu của chính mình để xác thực họ là ai. Nếu họ đã được ủy quyền trong cấu hình `sudoers` để chạy lệnh đó với đặc quyền với các tùy chọn được cho phép, lệnh đó sẽ hoạt động.

TIP

Theo mặc định, lệnh `sudo` được xác thực đầu tiên sẽ xác thực các lệnh `sudo` tiếp theo trong một khoảng thời gian rất ngắn. Điều này có thể được cấu hình bởi quản trị viên hệ thống.

Tập Kiểm soát Truy cập

Gần như tất cả các hệ điều hành đều có một tập hợp các vị trí được sử dụng để lưu trữ kiểm soát

về truy cập. Trong Linux, đây thường là các tệp văn bản nằm trong thư mục `/etc`, tức nơi lưu trữ các tệp cấu hình hệ thống. Theo mặc định, thư mục này có thể được đọc bởi mọi người dùng trên hệ thống nhưng chỉ có thể được ghi bởi root.

Các tệp chính liên quan đến tài khoản người dùng, thuộc tính và kiểm soát truy cập là:

`/etc/passwd`

Tệp này lưu trữ thông tin cơ bản về người dùng trên hệ thống, bao gồm UID và GID, thư mục chính, vỏ, v.v. Mặc dù có tên như vậy nhưng không có mật khẩu nào được lưu trữ ở đây.

`/etc/group`

Tệp này lưu trữ thông tin cơ bản về tất cả các nhóm người dùng trên hệ thống như tên nhóm, GID và các thành viên.

`/etc/shadow`

Đây là nơi lưu trữ mật khẩu người dùng. Chúng sẽ được băm nhỏ vì mục đích bảo mật.

`/etc/gshadow`

Tệp này lưu trữ thông tin chi tiết hơn về các nhóm, bao gồm mật khẩu đã băm nhỏ cho phép người dùng tạm thời trở thành thành viên của nhóm, danh sách người dùng có thể trở thành thành viên của nhóm tại bất cứ thời điểm nào và danh sách của người quản trị nhóm.

WARNING

Các tệp này không được thiết kế để và không bao giờ nên được chỉnh sửa trực tiếp. Bài học này chỉ nói đến các thông tin được lưu trữ trong các tệp này và không nói về việc chỉnh sửa các tệp này.

Theo mặc định, mọi người dùng đều có thể nhập `/etc` và đọc các tệp `/etc/passwd` và `/etc/group`. Và cũng theo mặc định, không người dùng nào (ngoại trừ root) có thể đọc các tệp `/etc/shadow` hoặc `/etc/gshadow`.

Ngoài ra còn có các tệp liên quan đến việc nâng đặc quyền cơ bản trên các hệ thống Linux như trên các lệnh `su` và `sudo`. Theo mặc định, những lệnh này chỉ có thể được truy cập bởi người dùng root.

`/etc/sudoers`

Tệp này kiểm soát việc ai có thể sử dụng lệnh `sudo` và bằng cách nào.

`/etc/sudoers.d`

Thư mục này có thể chứa các tệp bổ sung cho cài đặt trên tệp `sudoers`.

Theo như kỳ thi LPI Linux Essentials, ta chỉ cần biết đường dẫn và tên tệp của tệp cấu hình `sudo` mặc định là `/etc/sudoers`. Cấu hình của nó nằm ngoài phạm vi của những tài liệu này.

WARNING

Mặc dù `/etc/sudoers` là một tệp văn bản, nó không bao giờ nên được chỉnh sửa trực tiếp. Nếu cần bất kỳ thay đổi nào đối với nội dung của nó, chúng phải được thực hiện bằng tiện ích `visudo`.

/etc/passwd

Tệp `/etc/passwd` thường được gọi là “tệp mật khẩu”. Mỗi dòng trong tệp có chứa nhiều trường luôn được phân định bằng dấu hai chấm (:). Mặc dù có tên như vậy, hiện nay các hàm băm mật khẩu một chiều thực tế không được lưu trữ trong tệp này.

Cú pháp điển hình của một dòng trong tệp này là:

```
USERNAME:PASSWORD:UID:GID:GECOS:HOMEDIR:SHELL
```

Và trong đó:

USERNAME

Tên người dùng - hay còn gọi là thông tin đăng nhập (tên) - như `root`, `nobody`, `emma`.

PASSWORD

Vị trí kế thừa của mật khẩu băm nhỏ. Hầu như luôn luôn là `x` và cho biết rằng mật khẩu được lưu trữ trong tệp `/etc/shadow`.

UID

ID người dùng (UID), như `0`, `99`, `1024`.

GID

ID nhóm mặc định (GID), như `0`, `99`, `1024`.

GECOS

Một danh sách giá trị phân tách bằng dấu phẩy chứa thông tin người dùng bao gồm tên, vị trí, số điện thoại. Ví dụ: `Emma Smith,42 Douglas St,555.555.5555`

HOMEDIR

Đường dẫn đến thư mục chính của người dùng, như `/root`, `/home/emma`, v.v.

SHELL

Vỏ mặc định cho người dùng này, như `/bin/bash`, `/sbin/nologin`, `/bin/ksh`, v.v.

Ví dụ: dòng sau mô tả người dùng `emma`:

```
emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555:/home/emma:/bin/bash
```

Hiểu về Trường GECOS

Trường GECOS chứa 3 trường trở lên được phân tách bằng dấu phẩy (,) - hay còn gọi là danh sách *Giá trị được phân tách bằng dấu phẩy* (CSV). Mặc dù không có tiêu chuẩn bắt buộc, các trường thường đều sẽ tuân theo thứ tự sau:

```
NAME, LOCATION, CONTACT
```

Trong đó:

NAME

là “Tên đầy đủ” của người dùng hoặc “Tên phần mềm” trong trường hợp là tài khoản dịch vụ.

LOCATION

thường là vị trí vật lý của người dùng trong tòa nhà, số phòng hoặc bộ phận hoặc người liên hệ trong trường hợp đó là tài khoản dịch vụ.

CONTACT

liệt kê thông tin liên lạc như số điện thoại nhà riêng hoặc cơ quan.

Các trường bổ sung có thể bao gồm thông tin liên hệ bổ sung, chẳng hạn như số nhà hoặc địa chỉ email. Để thay đổi thông tin trên trường GECOS, hãy sử dụng lệnh `chfn` và trả lời các câu hỏi như bên dưới. Nếu không có tên người dùng nào được cung cấp sau tên lệnh, bạn sẽ thay đổi thông tin cho người dùng hiện tại:

```
$ chfn
```

```
Changing the user information for emma
Enter the new value, or press ENTER for the default
Full Name: Emma Smith
Room Number []: 42
Work Phone []: 555.555.5555
Home Phone []: 555.555.6666
```

/etc/group

Tập `/etc/group` chứa các trường luôn được phân tách bằng dấu hai chấm (:) và lưu trữ thông tin

cơ bản về các nhóm trên hệ thống. Đôi khi nó còn được gọi là “tệp nhóm”. Cú pháp cho mỗi dòng là:

```
NAME:PASSWORD:GID:MEMBERS
```

Trong đó:

NAME

là tên nhóm, như `root`, `users`, `emma`, v.v.

PASSWORD

vị trí kế thừa của mật khẩu bấm nhỏ của nhóm tùy chọn. Hầu như luôn luôn là `x`, cho biết rằng mật khẩu (nếu được xác định) được lưu trữ trong tệp `/etc/gshadow`.

GID

ID nhóm (GID), như `0`, `99`, `1024`.

MEMBERS

danh sách được phân tách bằng dấu phẩy của tên những người dùng là thành viên của nhóm, chẳng hạn như `jsmith`, `emma`.

Ví dụ bên dưới hiển thị một dòng chứa thông tin về nhóm `students`:

```
students:x:1023:jsmith,emma
```

Người dùng không cần phải được liệt kê trong trường thành viên khi nhóm là nhóm chính cho người dùng. Nếu người dùng được liệt kê thì cũng là thừa — tức là không có thay đổi về chức năng dù có được liệt kê hay không.

NOTE

Việc sử dụng mật khẩu cho các nhóm nằm ngoài phạm vi của phần này; tuy nhiên, nếu được xác định, mật khẩu bấm nhỏ sẽ được lưu trữ trong tệp `/etc/gshadow`. Điều này cũng nằm ngoài phạm vi của phần này.

/etc/shadow

Bảng sau đây sẽ liệt kê các thuộc tính được lưu trữ trong tệp `/etc/shadow`, thường được gọi là “tệp shadow”. Tệp này chứa các trường luôn được phân tách bằng dấu hai chấm (:). Mặc dù tệp có nhiều trường nhưng ngoại trừ hai trường đầu tiên thì hầu hết chúng đều nằm ngoài phạm vi của bài học này.

Cú pháp cơ bản cho một dòng trong tệp này là:

```
USERNAME:PASSWORD:LASTCHANGE:MINAGE:MAXAGE:WARN:INACTIVE:EXPDATE
```

Trong đó:

USERNAME

Tên người dùng (giống như `/etc/passwd`), như `root`, `nobody`, `emma`.

PASSWORD

Phần băm nhỏ một chiều của mật khẩu, bao gồm cả các muối trước đó. Ví dụ: `!!`, `!$1$01234567$ABC...`, `$6$012345789ABCDEF$012...`

LASTCHANGE

Ngày thay đổi mật khẩu cuối cùng tính bằng ngày kể từ điểm “kỷ nguyên”, chẳng hạn như 17909.

MINAGE

Tuổi mật khẩu tối thiểu tính bằng ngày.

MAXAGE

Tuổi mật khẩu tối đa tính bằng ngày.

WARN

Thời gian cảnh báo trước khi hết hạn mật khẩu, tính bằng ngày.

INACTIVE

Tuổi mật khẩu tối đa đã hết hạn, tính bằng ngày.

EXPDATE

Ngày hết hạn mật khẩu, tính bằng ngày kể từ “kỷ nguyên”.

Trong ví dụ bên dưới, bạn có thể thấy một mục nhập mẫu từ tệp `/etc/shadow`. Hãy lưu ý rằng có một số giá trị như `INACTIVE` và `EXPDATE` sẽ không được xác định.

```
emma:$6$nP532JDDogQYZF8I$bjFNh9eT1xpb9/n6pmj1Iwgu7hGjH/eytSdttbmVv0M1yTMFgBIXESFNumTo9EGxxH1  
0T1HGQzR0so4n1npbE0:18064:0:99999:7:::
```

“Kỷ nguyên” của hệ thống POSIX là nửa đêm (0000), Giờ Tọa độ Quốc tế (UTC), vào Thứ Năm, ngày 1 tháng 1 năm 1970. Hầu hết các ngày và giờ POSIX được tính bằng giây kể từ “kỷ nguyên” hoặc là

ngày kể từ “kỷ nguyên” theo trường hợp của tệp `/etc/shadow`.

NOTE

Tệp shadow được thiết kế để chỉ có thể được đọc bởi siêu người dùng và các dịch vụ xác thực hệ thống cốt lõi được chọn để kiểm tra hàm băm mật khẩu một chiều khi đăng nhập hoặc thời điểm xác thực khác.

Mặc dù có tồn tại các giải pháp xác thực khác nhau, phương pháp lưu trữ mật khẩu cơ bản là *chức năng băm* một chiều. Điều này được thực hiện để mật khẩu không bao giờ được lưu trữ dưới dạng văn bản rõ ràng trên hệ thống, vì hàm băm sẽ không thể bị đảo ngược. Bạn có thể băm mật khẩu nhưng (trên lý tưởng) không thể biến các phần băm đó trở lại thành mật khẩu ban đầu.

Trong trường hợp tối cần, cần có một phương pháp mạnh tay để băm tất cả các tổ hợp mật khẩu cho đến khi một kết hợp khớp. Để giảm thiểu vấn đề mã băm mật khẩu bị hoá giải trên một hệ thống, các hệ thống Linux sử dụng một “muối” ngẫu nhiên trên mỗi mã băm mật khẩu cho người dùng. Vì vậy, hàm băm cho mật khẩu người dùng trên các hệ thống Linux khác nhau sẽ không giống nhau ngay cả khi mật khẩu giống nhau.

Trong tệp `/etc/shadow`, mật khẩu có thể sẽ có nhiều dạng. Các hình thức này thường bao gồm các phần sau đây:

!!

Điều này có nghĩa là tài khoản “bị vô hiệu hóa” (không thể xác thực), không lưu trữ mật khẩu đã băm.

!\$1\$01234567\$ABC...

Tài khoản “bị vô hiệu hóa” (do dấu chấm than ban đầu) với hàm băm, muối băm và chuỗi băm trước đó được lưu trữ.

\$1\$0123456789ABC\$012...

Tài khoản “có hiệu lực” với hàm băm, muối băm và chuỗi băm được lưu trữ.

Hàm băm, muối băm và chuỗi băm được đặt trước và phân định bằng ký hiệu đô la (\$). Độ dài muối băm phải từ tám đến mười sáu (8-16) ký tự. Ví dụ về ba muối băm phổ biến nhất là:

\$1\$01234567\$ABC...

Hàm băm của MD5 (1), với độ dài hàm băm ví dụ là tám.

\$5\$01234567ABCD\$012...

Hàm băm của SHA256 (5), với độ dài hàm băm ví dụ là mười hai.

\$6\$01234567ABCD\$012...

Hàm băm của SHA512 (6), với độ dài hàm băm ví dụ là mười hai.

NOTE

Hàm băm MD5 được coi là không an toàn về mặt mật mã với mức ASIC ngày nay (từ những năm 2010 trở lại đây) và thậm chí là cả hiệu suất SIMD điện toán chung. Ví dụ: Tiêu chuẩn xử lý thông tin liên bang Hoa Kỳ (FIPS) không cho phép sử dụng MD5 cho bất kỳ chức năng mã hóa nào ngoài các khía cạnh xác thực rất hạn chế nhưng không phải tính toàn vẹn của chữ ký số hoặc các mục đích tương tự.

Từ khía cạnh của các mục tiêu và kỳ thi LPI Linux Essentials, ta chỉ cần hiểu các phần mật khẩu băm nhỏ cho người dùng cục bộ chỉ được lưu trữ trong tệp `/etc/shadow` và chỉ có các dịch vụ xác thực được chọn mới có thể đọc hoặc chỉ siêu người dùng mới có thể sửa đổi thông qua các lệnh khác .

Bài tập Hướng dẫn

1. Hãy xem xét đầu ra sau của lệnh `id`:

```
$ id emma
uid=1000(emma) gid=1000(emma)
groups=1000(emma),4(adm),5(tty),10(uucp),20(dialout),27(sudo),46(plugdev)
```

Các thuộc tính sau được lưu trữ trong tệp nào?

UID và GID	
Nhóm	

- Ngoài ra, mật khẩu người dùng được lưu trong tệp nào?

2. Loại mã hóa mật khẩu mặc định được sử dụng để lưu trữ cục bộ mật khẩu trên hệ thống Linux?

- Asymmetric
- One-way Hash
- Symmetric
- ROT13

3. Nếu một tài khoản có User ID (UID) được liệt kê dưới 1000 thì đây là loại tài khoản gì?

4. Làm cách nào để có thể nhận được danh sách các thông tin đăng nhập đang hoạt động trong hệ thống của mình và số lượng các thông tin đăng nhập đó?

5. Hãy sử dụng lệnh `grep` và nhận về kết quả bên dưới với thông tin về người dùng `emma`.

```
$ grep emma /etc/passwd
emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555,:/home/emma:/bin/ksh
```

Hãy điền thông tin thích hợp vào chỗ trống của biểu đồ sử dụng đầu ra của lệnh trước đó.

Username	
Password	
UID	
Primary GID	
GECOS	
Home Directory	
Shell	

Bài tập Mở rộng

1. Hãy so sánh kết quả của `last` với `w` và `who`. Những chi tiết nào bị thiếu trong mỗi lệnh khi so với nhau?

2. Hãy thử ra lệnh `who` và `w -his`.

- Thông tin nào đã bị xóa khỏi đầu ra của lệnh `w` với tùy chọn “no header” (`-h`) và “short” (`-s`)?

- Thông tin nào đã được thêm vào đầu ra của lệnh `w` với tùy chọn “địa chỉ ip” (`-i`)?

3. Tập nào là tập lưu trữ phần băm mật khẩu một chiều của tài khoản người dùng?

4. Tập nào chứa danh sách các nhóm mà tài khoản người dùng là thành viên trong đó? Logic nào có thể được sử dụng để biên soạn danh sách nhóm mà tài khoản người dùng là thành viên trong đó?

5. Theo mặc định, một hoặc nhiều (1+) các tập sau đây không thể được đọc bởi người dùng thông thường không có đặc quyền. Đó là những tập nào sau đây?

- `/etc/group`
- `/etc/passwd`
- `/etc/shadow`
- `/etc/sudoers`

6. Bạn sẽ thay đổi vỏ đăng nhập của người dùng hiện tại thành Vỏ Korn (`/usr/bin/ksh`) ở chế độ không tương tác như thế nào?

7. Tại sao thư mục chính của người dùng `root` không được đặt trong thư mục `/home`?

Tóm tắt

Trong bài học này, chúng ta đã khám phá cơ sở dữ liệu nhóm và người dùng Linux. Chúng ta đã biết các thuộc tính quan trọng nhất của người dùng và nhóm, bao gồm tên và ID bằng chữ số của chúng. Chúng ta cũng đã tìm hiểu cách băm mật khẩu hoạt động trên Linux và cách người dùng được chỉ định vào các nhóm.

Tất cả thông tin này đều được lưu trữ trong bốn tệp sau; các tệp này cung cấp việc kiểm soát truy cập bảo mật cục bộ cơ bản nhất trên hệ thống Linux:

/etc/passwd

Tất cả các thuộc tính POSIX của tài khoản người dùng cục bộ hệ thống (ngoại trừ băm mật khẩu); có thể được đọc bởi mọi người dùng.

/etc/group

Tất cả các thuộc tính POSIX của tài khoản nhóm cục bộ hệ thống; có thể được đọc bởi mọi người dùng.

/etc/shadow

Tất cả các phần băm mật khẩu của người dùng cục bộ trên hệ thống (và thông tin hết hạn); không thể được đọc bởi bất kỳ người dùng nào (ngoài các quy trình được chọn).

/etc/sudoers

Tất cả các thông tin/ sự cho phép về vấn đề nâng đặc quyền cục bộ hệ thống bằng lệnh `sudo`.

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

id

Liệt kê ID người dùng và nhóm thực (hoặc hiệu quả).

last

Liệt kê những người dùng đăng nhập cuối cùng

who

Liệt kê những người dùng hiện đang đăng nhập.

w

Tương tự như `who` nhưng có ngữ cảnh bổ sung.

su

Chuyển sang người dùng khác bằng vỏ đăng nhập hoặc chạy các lệnh với tư cách người dùng

đó bằng cách chuyển mật khẩu của người dùng đó.

sudo

Chuyển đổi Người dùng (hoặc Siêu Người dùng); nếu được phép, người dùng hiện tại nhập mật khẩu của chính họ (nếu được yêu cầu) để nâng cao đặc quyền.

chsh

Thay đổi vỏ của người dùng.

chfn

Thay đổi thông tin của người dùng trên trường GECOS.

Đáp án Bài tập Hướng dẫn

1. Hãy xem xét đầu ra sau của lệnh `id`

```
$ id emma
uid=1000(emma) gid=1000(emma)
groups=1000(emma),4(adm),5(tty),10(uucp),20(dialout),27(sudo),46(plugdev)
```

Các thuộc tính sau được lưu trữ trong tệp nào?

UID và GID	<code>/etc/passwd</code>
Nhóm	<code>/etc/group</code>

◦ Ngoài ra, mật khẩu người dùng được lưu trong tệp nào?

Mật khẩu người dùng đã băm được lưu trữ trong `/etc/shadow`.

2. Loại mã hóa mật khẩu mặc định được sử dụng để lưu trữ cục bộ mật khẩu trên hệ thống Linux?

Theo mặc định, hàm băm một chiều được sử dụng để lưu trữ mật khẩu.

3. Nếu một tài khoản có User ID (UID) được liệt kê dưới 1000 thì đây là loại tài khoản gì?

Tài khoản có UID thấp hơn 1000 thường là tài khoản hệ thống.

4. Làm cách nào để có thể nhận được danh sách các thông tin đăng nhập đang hoạt động trong hệ thống của mình và số lượng các thông tin đăng nhập đó?

Sử dụng lệnh `w`. Bên cạnh danh sách tất cả các lần đăng nhập đang hoạt động, nó cũng sẽ hiển thị thông tin về số lượng người dùng đã đăng nhập cùng với tải trọng hệ thống và thời gian hoạt động.

5. Hãy sử dụng lệnh `grep` và nhận về kết quả bên dưới với thông tin về người dùng `emma`.

```
$ grep emma /etc/passwd
emma:x:1000:1000:Emma Smith,42 Douglas St,555.555.5555,:/home/emma:/bin/ksh
```

Hãy điền thông tin thích hợp vào chỗ trống của biểu đồ sử dụng đầu ra của lệnh trước đó.

Username	emma
Password	x - phải luôn là x đối với thông tin đăng nhập của người dùng đang hoạt động và hợp lệ
UID	1000
Primary GID	1000
GECOS	Emma Smith,42 Douglas St,555.555.5555
Home Directory	/home/emma
Shell	/bin/ksh

Đáp án Bài tập Mở rộng

1. Hãy so sánh kết quả của `last` với `w` và `who`. Những chi tiết nào bị thiếu trong mỗi lệnh khi so với nhau?

Công cụ `w` và `who` chỉ liệt kê những người dùng hiện tại đã đăng nhập vào hệ thống, trong khi `last` còn liệt kê cả những người dùng đã ngắt kết nối. Lệnh `w` sẽ liệt kê việc sử dụng hệ thống, trong khi lệnh `who` thì không.

2. Hãy thử ra lệnh `who` và `w -his`.

- Thông tin nào đã bị xóa khỏi đầu ra của lệnh `w` với tùy chọn “no header” (`-h`) và “short” (`-s`)?

Đầu ra sẽ không có chủ đề; điều này hữu ích trong việc phân tích cú pháp; thời gian đăng nhập cũng như thông tin chọn CPU không được liệt kê.

- Thông tin nào đã được thêm vào đầu ra của lệnh `w` với tùy chọn “ip address” (`-i`)?

Lệnh này in địa chỉ IP, thay vì thử phân giải DNS, sẽ in tên máy chủ. Tùy chọn này giúp đầu ra của lệnh `w` phù hợp hơn với đầu ra mặc định của lệnh `last`.

3. Tập nào là tập lưu trữ hàm băm mật khẩu một chiều của tài khoản người dùng?

Tập `/etc/shadow` lưu trữ hàm băm mật khẩu một chiều của tài khoản người dùng vì nó không thể được đọc bằng tài khoản người dùng thông thường không có đặc quyền, không giống như tập `/etc/passwd`.

4. Tập nào chứa danh sách các nhóm mà tài khoản người dùng là thành viên trong đó? Logic nào có thể được sử dụng để biên soạn danh sách nhóm mà tài khoản người dùng là thành viên trong đó?

Tập `/etc/group` có danh sách tên người dùng CSV trong trường cuối cùng, tức “thành viên”, của bất kỳ dòng nào cho một nhóm.

Bất kỳ dòng nào trong tập `/etc/group` mà trong đó người dùng được liệt kê trong trường cuối cùng (“members”) đều có nghĩa là người dùng là thành viên của nhóm đó — giả sử nhóm đó được định dạng chính xác (được phân tách bằng CSV). Ngoài ra, tư cách thành viên nhóm chính của người dùng trong tập `/etc/passwd` cũng sẽ có một mục trùng trong tập `/etc/group` cho cả tên nhóm và GID.

5. Theo mặc định, một hoặc nhiều (1+) các tập sau đây không thể được đọc bởi người dùng thông thường không có đặc quyền. Đó là những tập nào sau đây?

- `/etc/group`
- `/etc/passwd`
- `/etc/shadow`
- `/etc/sudoers`

Các tệp `/etc/shadow` và `/etc/sudoers` không thể được đọc theo mặc định ngoại trừ bởi các dịch vụ được chọn hoặc siêu người dùng. Những câu trả lời này sẽ được tùy chỉnh dựa trên hệ thống và tên người dùng được sử dụng trong phòng máy.

6. Bạn sẽ thay đổi vỏ đăng nhập của người dùng hiện tại thành Vỏ Korn (`/usr/bin/ksh`) ở chế độ không tương tác như thế nào?

```
$ chsh -s /usr/bin/ksh
```

7. Tại sao thư mục chính của người dùng `root` không được đặt trong thư mục `/home`?

Bởi vì cần có tài khoản `root` để khắc phục sự cố và sửa lỗi, có thể bao gồm cả các sự cố hệ thống tệp liên quan đến thư mục `/home`. Trong những trường hợp như vậy, `root` phải hoạt động với đầy đủ chức năng ngay cả khi hệ thống tệp `/home` chưa khả dụng.



5.2 Tạo người dùng và nhóm

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 5.2](#)

Khối lượng

2

Các lĩnh vực kiến thức chính

- Lệnh người dùng và nhóm
- ID người dùng

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

- `/etc/passwd`, `/etc/shadow`, `/etc/group`, `/etc/skel/`
- `useradd`, `groupadd`
- `passwd`



5.2 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	5 Bảo mật và Quyền trong Tập
Mục tiêu:	5.2 Tạo Người dùng và Nhóm
Bài:	1 trên 1

Giới thiệu

Việc quản lý người dùng và nhóm trên một máy Linux là một trong những khía cạnh chính của tác vụ quản trị hệ thống. Trên thực tế, Linux là một hệ điều hành đa người dùng mà trong đó nhiều người dùng có thể sử dụng cùng một máy trong cùng một thời điểm.

Thông tin về người dùng và nhóm được lưu trữ trong bốn tệp trong cây thư mục `/etc/`:

`/etc/passwd`

một tệp gồm bảy trường được phân tách bằng dấu hai chấm chứa thông tin cơ bản về người dùng

`/etc/group`

một tệp gồm bốn trường được phân tách bằng dấu hai chấm chứa thông tin cơ bản về nhóm

`/etc/shadow`

một tệp gồm chín trường được phân tách bằng dấu hai chấm chứa mật khẩu người dùng đã được mã hóa

/etc/gshadow

tệp gồm bốn tệp trường được phân tách bằng dấu hai chấm chứa mật khẩu nhóm đã được mã hóa

Tất cả các tệp này đều được cập nhật bởi một bộ công cụ dòng lệnh dùng để quản lý người dùng và nhóm; chúng ta sẽ thảo luận sau trong bài học này. Chúng cũng có thể được quản lý bằng các ứng dụng đồ họa (riêng cho từng bản phân phối Linux) cung cấp các giao diện quản lý đơn giản và trực tiếp hơn.

WARNING

Mặc dù các tệp đều là văn bản thuần túy nhưng ta không được chỉnh sửa chúng trực tiếp. Hãy luôn sử dụng các công cụ được cung cấp cùng với bản phân phối của bạn cho mục đích này.

Tệp /etc/passwd

/etc/passwd là một tệp luôn có thể đọc được chứa danh sách người dùng, mỗi người dùng trên một dòng riêng biệt:

```
frank:x:1001:1001:./home/frank:/bin/bash
```

Mỗi dòng bao gồm bảy trường được phân tách bằng dấu hai chấm:

Tên người dùng

Tên được sử dụng khi người dùng đăng nhập vào hệ thống.

Mật khẩu

Mật khẩu được mã hóa (hoặc x nếu mật khẩu ẩn được sử dụng).

ID người dùng (UID)

Số ID được gán cho người dùng trong hệ thống.

ID nhóm (GID)

Số nhóm chính của người dùng trong hệ thống.

GECOS

Trường nhận xét tùy chọn, được sử dụng để thêm thông tin bổ sung về người dùng (chẳng hạn như tên đầy đủ). Trường có thể chứa nhiều mục được phân tách bằng dấu phẩy.

Thư mục chính

Đường dẫn tuyệt đối của thư mục chính của người dùng.

Vỏ

Đường dẫn tuyệt đối của chương trình được khởi chạy tự động khi người dùng đăng nhập vào hệ thống (thường là vỏ tương tác, chẳng hạn như `/bin/bash`).

Tệp `/etc/group`

`/etc/group` là một tệp luôn có thể đọc được chứa danh sách các nhóm, mỗi nhóm nằm trên một dòng riêng biệt:

```
developer:x:1002:
```

Mỗi dòng bao gồm bốn trường được phân tách bằng dấu hai chấm:

Tên nhóm

Tên của nhóm.

Mật khẩu nhóm

Mật khẩu được mã hóa của nhóm (hoặc `x` nếu sử dụng mật khẩu ẩn).

ID nhóm (GID)

Số ID được gán cho nhóm trong hệ thống.

Danh sách thành viên

Danh sách người dùng thuộc nhóm được phân tách bằng dấu phẩy, ngoại trừ những người dùng thuộc nhóm chính.

Tệp `/etc/shadow`

`/etc/shadow` là một tệp chỉ có thể được đọc bởi người dùng gốc và người dùng có đặc quyền hệ thống; nó chứa mật khẩu được mã hóa của người dùng, mỗi mật khẩu ở trên một dòng riêng biệt:

```
frank:$6$i9gjM4Md4Mue1ZCd$7jJa8Cd2bbADFH4dwtfvTvJLOYCCCBf/.jYbK1IMYx7Wh4fErXcc2xQVU2N1gb97yI  
YaiqH.jjJammzof2Jfr/:18029:0:99999:7:::
```

Mỗi dòng bao gồm chín trường được phân tách bằng dấu hai chấm:

Tên tài khoản

Tên được sử dụng khi người dùng đăng nhập vào hệ thống.

Mật khẩu được mã hóa

Mật khẩu được mã hóa của người dùng (nếu giá trị là ! tức là tài khoản đã bị khóa).

Ngày thay đổi mật khẩu cuối cùng

Ngày thay đổi mật khẩu cuối cùng, tức số ngày kể từ ngày 01/01/1970. Giá trị 0 có nghĩa là người dùng phải thay đổi mật khẩu ở lần truy cập tiếp theo.

Tuổi mật khẩu tối thiểu

Số ngày tối thiểu sau khi thay đổi mật khẩu mà người dùng phải trải qua trước khi được phép thay đổi mật khẩu lần tiếp theo.

Tuổi mật khẩu tối đa

Số ngày tối đa phải trôi qua trước khi yêu cầu thay đổi mật khẩu.

Thời gian cảnh báo mật khẩu

Số ngày trước khi mật khẩu hết hạn mà trong khoảng thời gian đó người dùng được cảnh báo phải thay đổi mật khẩu.

Thời gian mật khẩu không hoạt động

Số ngày sau khi mật khẩu hết hạn mà người dùng nên cập nhật mật khẩu. Sau thời gian này, nếu người dùng không thay đổi mật khẩu, tài khoản sẽ bị vô hiệu hóa.

Ngày hết hạn tài khoản

Ngày được tính kể từ ngày 01/01/1970, mà trong đó tài khoản người dùng sẽ bị vô hiệu hóa. Trường trống có nghĩa là tài khoản người dùng sẽ không bao giờ hết hạn.

Một trường dành riêng

Một trường được dành riêng để sử dụng trong tương lai.

Tệp `/etc/gshadow`

`/etc/gshadow` là một tệp chỉ có thể đọc được bởi người dùng gốc và bởi người dùng có quyền gốc chứa mật khẩu được mã hóa cho các nhóm, mỗi nhóm ở một dòng riêng biệt:

```
developer:$6$7QUIhUX1Wd06$H7k0YgsboLkDseFHpk041wAtweSUQHipoxIgo83QNDxYtYwgmZTCU0qSCuCkErmyR2  
63rvHiLctZVDR7Ya9Ai1::
```

Mỗi dòng bao gồm bốn trường được phân tách bằng dấu hai chấm:

Tên nhóm

Tên của nhóm.

Mật khẩu được mã hóa

Mật khẩu được mã hóa cho nhóm (mật khẩu này được sử dụng khi một người dùng không phải là thành viên của nhóm muốn tham gia nhóm bằng cách sử dụng lệnh `newgrp` — nếu mật khẩu bắt đầu bằng `!` thì không ai được phép truy cập nhóm bằng `newgrp`).

Quản trị viên nhóm

Danh sách quản trị viên của nhóm được phân tách bằng dấu phẩy (họ có thể thay đổi mật khẩu của nhóm và có thể thêm hoặc xóa thành viên nhóm bằng lệnh `gpasswd`).

Thành viên nhóm

Danh sách các thành viên của nhóm được phân tách bằng dấu phẩy.

Bây giờ, chúng ta đã biết về nơi lưu trữ thông tin người dùng và nhóm. Hãy cùng tìm hiểu về các công cụ dòng lệnh quan trọng nhất để cập nhật các tệp này.

Thêm và Xóa Tài khoản Người dùng

Trong Linux, ta có thể thêm tài khoản người dùng mới bằng lệnh `useradd` và xóa tài khoản người dùng bằng lệnh `userdel`.

Nếu bạn muốn tạo tài khoản người dùng mới có tên `frank` với cài đặt mặc định, bạn có thể chạy lệnh như sau:

```
# useradd frank
```

Sau khi tạo người dùng mới, bạn có thể đặt mật khẩu bằng `passwd`:

```
# passwd frank
Changing password for user frank.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

Cả hai lệnh này đều yêu cầu quyền gốc. Khi bạn chạy lệnh `useradd`, thông tin người dùng và nhóm được lưu trữ trong cơ sở dữ liệu mật khẩu và nhóm sẽ được cập nhật cho tài khoản người dùng mới được tạo và nếu được chỉ định, thư mục chính của người dùng mới sẽ được tạo cùng với một nhóm có cùng tên với tài khoản người dùng.

Hãy nhớ rằng bạn luôn có thể sử dụng tiện ích `grep` để lọc cơ sở dữ liệu nhóm và mật khẩu để chỉ hiển thị mục đề cập đến một người dùng hoặc nhóm cụ thể. Đối với ví dụ trên, bạn có thể sử dụng

TIP

```
cat /etc/passwd | grep frank
```

```
hoặc grep frank /etc/passwd
```

để xem thông tin cơ bản về tài khoản `frank` mới được tạo.

Các tùy chọn quan trọng nhất áp dụng cho lệnh `useradd` là:

-c

Tạo tài khoản người dùng mới với tùy chỉnh (ví dụ: tên đầy đủ).

-d

Tạo tài khoản người dùng mới với thư mục chính tùy chỉnh.

-e

Tạo tài khoản người dùng mới bằng cách đặt ngày cụ thể mà tài khoản sẽ bị vô hiệu hóa.

-f

Tạo tài khoản người dùng mới bằng cách đặt số ngày sau khi mật khẩu hết hạn mà trong khoảng thời gian đó người dùng nên cập nhật mật khẩu.

-g

Tạo tài khoản người dùng mới với một GID cụ thể

-G

Tạo tài khoản người dùng mới bằng cách thêm nó vào nhiều nhóm phụ.

-m

Tạo một tài khoản người dùng mới với thư mục chính của nó.

-M

Tạo tài khoản người dùng mới không có thư mục chính.

-s

Tạo tài khoản người dùng mới với vỏ đăng nhập cụ thể.

-u

Tạo tài khoản người dùng mới với UID cụ thể. Khi tài khoản người dùng mới được tạo, bạn có

thể sử dụng các lệnh `id` và `groups` để tìm ra UID, GID và các nhóm mà tài khoản đó thuộc về.

```
# id frank
uid=1000(frunk) gid=1000(frunk) groups=1000(frunk)
# groups frank
frank : frank
```

TIP

Hãy nhớ kiểm tra và chỉnh sửa tệp `/etc/login.defs`; tệp này sẽ xác định các tham số cấu hình kiểm soát việc tạo người dùng và nhóm. Ví dụ: bạn có thể đặt phạm vi UID và GID có thể được chỉ định cho tài khoản nhóm và người dùng mới để cho biết rằng bạn không cần sử dụng tùy chọn `-m` để tạo thư mục chính của người dùng mới và nếu hệ thống cần phải tự động tạo một nhóm mới cho mỗi người dùng mới.

Nếu muốn xóa tài khoản người dùng, bạn có thể sử dụng lệnh `userdel`. Cụ thể, lệnh này sẽ cập nhật thông tin được lưu trữ trong cơ sở dữ liệu tài khoản, xóa tất cả các mục liên quan đến người dùng được chỉ định. Tùy chọn `-r` cũng sẽ xóa thư mục chính của người dùng và tất cả các nội dung của nó cùng với bộ đệm thư của người dùng. Các tệp khác nằm ở các vị trí khác phải được tìm kiếm và xóa bằng cách thủ công.

```
# userdel -r frank
```

Giống như trước, bạn cần quyền gốc để xóa tài khoản người dùng.

Thư mục Skeleton

Khi bạn thêm tài khoản người dùng mới, thậm chí tạo cả thư mục chính cho nó, thư mục chính mới được tạo sẽ chứa các tệp và thư mục được sao chép từ thư mục skeleton (theo mặc định `/etc/skel`). Ý tưởng đằng sau điều này rất đơn giản: quản trị viên hệ thống muốn thêm người dùng mới có cùng tệp và thư mục vào chính của họ. Do đó, nếu bạn muốn tùy chỉnh các tệp và thư mục được tạo tự động trong thư mục chính của tài khoản người dùng mới, bạn phải thêm các tệp và thư mục mới này vào thư mục skeleton.

TIP

Hãy lưu ý rằng các tệp hồ sơ thường được tìm thấy trong thư mục skeleton là các tệp ẩn. Do đó, nếu bạn muốn liệt kê tất cả các tệp và thư mục trong thư mục skeleton mà sẽ được sao chép vào thư mục chính của người dùng mới được tạo, bạn phải sử dụng lệnh `ls -Al`.

Thêm và Xóa Nhóm

Đối với tác vụ quản lý nhóm, bạn có thể thêm hoặc xóa nhóm bằng lệnh `groupadd` và `groupdel`.

Nếu muốn tạo một nhóm mới có tên `developer`, bạn có thể chạy lệnh sau với quyền gốc:

```
# groupadd -g 1090 developer
```

Tùy chọn `-g` của lệnh này sẽ tạo một nhóm có GID cụ thể.

Nếu muốn xóa nhóm `developer`, bạn có thể chạy lệnh như sau:

```
# groupdel developer
```

WARNING

Hãy nhớ rằng khi thêm một tài khoản người dùng mới, nhóm chính và các nhóm phụ chứa tài khoản đó phải tồn tại trước khi khởi chạy lệnh `useradd`. Ngoài ra, bạn không thể xóa một nhóm nếu đó là nhóm chính của tài khoản người dùng.

Lệnh `passwd`

Lệnh này chủ yếu được sử dụng để thay đổi mật khẩu của người dùng. Bất kỳ người dùng nào cũng có thể thay đổi mật khẩu của họ, nhưng chỉ người dùng gốc mới có thể thay đổi mật khẩu của bất kỳ một người dùng nào.

Tùy thuộc vào tùy chọn `passwd` được sử dụng, bạn có thể kiểm soát các khía cạnh cụ thể của tuổi thọ của mật khẩu:

-d

Xóa mật khẩu của tài khoản người dùng (từ đó đặt mật khẩu trống, biến nó thành tài khoản không có mật khẩu).

-e

Buộc tài khoản người dùng thay đổi mật khẩu.

-l

Khóa tài khoản người dùng (mật khẩu được mã hóa có tiền tố là dấu chấm than).

-u

Mở khóa tài khoản người dùng (sẽ xóa dấu chấm than).

-S

Xuất thông tin về trạng thái mật khẩu cho một tài khoản cụ thể.

Các tùy chọn này chỉ có sẵn cho người dùng gốc. Để xem danh sách đầy đủ các tùy chọn, hãy tham khảo các trang hướng dẫn.

Bài tập Hướng dẫn

1. Đối với mỗi mục sau đây, hãy chỉ ra tệp mà nó đang nhắc đến:

- `developer:x:1010:frank,grace,dave`

- `root:x:0:0:root:/root:/bin/bash`

- `henry:1.AbCdEfGh123456789A1b2C3d4.:18015:20:90:5:30::`

- `henry:x:1000:1000:User Henry:/home/henry:/bin/bash`

- `staff:!:dave:carol,emma`

2. Hãy quan sát kết quả sau để trả lời bảy câu hỏi tiếp theo:

```
# cat /etc/passwd | tail -3
dave:x:1050:1050:User Dave:/home/dave:/bin/bash
carol:x:1051:1015:User Carol:/home/carol:/bin/sh
henry:x:1052:1005:User Henry:/home/henry:/bin/tcsh
# cat /etc/group | tail -3
web_admin:x:1005:frank,emma
web_developer:x:1010:grace,kevin,christian
dave:x:1050:
# cat /etc/shadow | tail -3
dave:$6$AbCdEfGh123456789A1b2C3D4e5F6G7h8i9:0:20:90:7:30::
carol:$6$q1w2e3r4t5y6u7i8AbCdEfGhIjKlMnOpQrStu:18015:0:60:7:::
henry:!$6$123456789aBcDeFgHa1B2c3d4E5f6g7H8I9:18015:0:20:5:::
# cat /etc/gshadow | tail -3
web_admin:!:frank:frank,emma
web_developer:!:kevin:grace,kevin,christian
dave:!::
```

- ID người dùng (UID) và ID nhóm (GID) của `carol` là gì?

- Vỏ nào được đặt cho `dave` và `henry`?

-
- Tên nhóm chính của `henry` là gì?

-
- Các thành viên của nhóm `web_developer` là ai? Ai trong số đó là quản trị viên nhóm?

-
- Người dùng nào không đăng nhập được vào hệ thống?

-
- Người dùng nào nên thay đổi mật khẩu trong lần đăng nhập tiếp theo vào hệ thống?

-
- Phải qua bao nhiêu ngày trước khi phải thay đổi mật khẩu cho `carol`?
-

Bài tập Mở rộng

1. Với quyền gốc, hãy chạy lệnh `useradd -m dave` để thêm tài khoản người dùng mới. Lệnh này thực hiện những thao tác nào? Giả sử rằng `CREATE_HOME` và `USERGROUPS_ENAB` trong `/etc/login.defs` được đặt là "yes".

2. Bây giờ, bạn đã tạo tài khoản `dave`; người dùng này có thể đăng nhập vào hệ thống không?

3. Hãy xác định ID người dùng (UID) và ID nhóm (GID) của `dave` và tất cả các thành viên của nhóm `dave`.

4. Hãy tạo các nhóm `sys_admin`, `web_admin` và `db_admin` và xác định ID nhóm (GID) của chúng.

5. Hãy thêm tài khoản người dùng mới có tên `carol` với UID 1035 và đặt `sys_admin` làm nhóm chính, `web_admin` và `db_admin` làm nhóm phụ.

6. Hãy xóa tài khoản người dùng `dave` và `carol` cũng như các nhóm `sys_admin`, `web_admin` và `db_admin` mà bạn đã tạo trước đó.

7. Hãy chạy lệnh `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` và mô tả đầu ra mà nó cung cấp cho bạn về quyền đối với tệp. Giả sử trong trường hợp hệ thống của bạn sử dụng mật khẩu ẩn, tệp nào trong số bốn tệp này bị ẩn vì lý do bảo mật?

8. Hãy chạy lệnh `ls -l /usr/bin/passwd`. Bit đặc biệt nào được thiết lập và ý nghĩa của nó là gì?

Tóm tắt

Trong bài học này, bạn đã học về:

- Nguyên tắc cơ bản của việc quản lý người dùng và nhóm trong Linux
- Quản lý thông tin người dùng và nhóm được lưu trữ trong cơ sở dữ liệu mật khẩu và nhóm
- Quản lý thư mục skeleton
- Thêm và xóa tài khoản người dùng
- Thêm và xóa tài khoản nhóm
- Thay đổi mật khẩu của tài khoản người dùng

Các lệnh sau đã được nhắc tới trong bài học này: `useradd`:: Tạo một tài khoản người dùng mới.

`groupadd`

Tạo một tài khoản nhóm mới.

`userdel`

Xóa tài khoản người dùng.

`groupdel`

Xóa một tài khoản nhóm.

`passwd`

Thay đổi mật khẩu của tài khoản người dùng và kiểm soát tất cả các khía cạnh về tuổi thọ mật khẩu.

Đáp án Bài tập Hướng dẫn

1. Đối với mỗi mục sau đây, hãy chỉ ra tệp mà nó đang nhắc đến:

- `developer:x:1010:frank,grace,dave`

`/etc/group`

- `root:x:0:0:root:/root:/bin/bash`

`/etc/passwd`

- `henry:1.AbCdEfGh123456789A1b2C3d4.:18015:20:90:5:30::`

`/etc/shadow`

- `henry:x:1000:1000:User Henry:/home/henry:/bin/bash`

`/etc/passwd`

- `staff:!:dave:carol,emma`

`/etc/gshadow`

2. Hãy quan sát kết quả sau để trả lời bảy câu hỏi tiếp theo:

```
# cat /etc/passwd | tail -3
dave:x:1050:1050:User Dave:/home/dave:/bin/bash
carol:x:1051:1015:User Carol:/home/carol:/bin/sh
henry:x:1052:1005:User Henry:/home/henry:/bin/tcsh
# cat /etc/group | tail -3
web_admin:x:1005:frank,emma
web_developer:x:1010:grace,kevin,christian
dave:x:1050:
# cat /etc/shadow | tail -3
dave:$6$AbCdEfGh123456789A1b2C3D4e5F6G7h8i9:0:20:90:7:30::
carol:$6$q1w2e3r4t5y6u7i8AbCdEfGhIjKlMnOpQrStu:18015:0:60:7:::
henry:!$6$123456789aBcDeFgHa1B2c3d4E5f6g7H8I9:18015:0:20:5:::
# cat /etc/gshadow | tail -3
web_admin:!:frank:frank,emma
web_developer:!:kevin:grace,kevin,christian
dave:!::
```

- ID người dùng (UID) và ID nhóm (GID) của `carol` là gì?

UID là 1051 và GID là 1015 (trường thứ ba và thứ tư trong `/etc/passwd`).

- Vỏ nào được đặt cho `dave` và `henry`?

`dave` sử dụng `/bin/bash` và `henry` sử dụng `/bin/tcsh` (trường thứ bảy trong `/etc/passwd`).

- Tên nhóm chính của `henry` là gì?

Tên nhóm là `web_admin` (trường đầu tiên trong `/etc/group`).

- Các thành viên của nhóm `web_developer` là ai? Ai trong số đó là quản trị viên nhóm?

Các thành viên là `grace`, `kevin` và `christian` (trường thứ tư trong `/etc/group`), nhưng chỉ có `kevin` là quản trị viên của nhóm (trường thứ ba trong `/etc/gshadow`).

- Người dùng nào không đăng nhập được vào hệ thống?

Tài khoản người dùng `henry` bị khóa (có dấu chấm than phía trước mật khẩu đã băm trong `/etc/shadow`).

- Người dùng nào nên thay đổi mật khẩu trong lần đăng nhập tiếp theo vào hệ thống?

Nếu trường thứ ba (Ngày thay đổi mật khẩu cuối cùng) trong `/etc/shadow` là 0 thì người dùng nên thay đổi mật khẩu của mình vào lần đăng nhập tiếp theo vào hệ thống. Do đó, `dave` phải thay đổi mật khẩu của mình.

- Phải qua bao nhiêu ngày trước khi phải thay đổi mật khẩu cho `carol`?

60 ngày (trường thứ năm trong `/etc/shadow`).

Đáp án Bài tập Mở rộng

1. Với quyền gốc, hãy chạy lệnh `useradd -m dave` để thêm tài khoản người dùng mới. Lệnh này thực hiện những thao tác nào? Giả sử rằng `CREATE_HOME` và `USERGROUPS_ENAB` trong `/etc/login.defs` được đặt là "yes".

Lệnh sẽ thêm một người dùng mới có tên `dave` vào danh sách người dùng trong hệ thống. Thư mục chính của `dave` sẽ được tạo (theo mặc định là `/home/dave`) và các tệp và thư mục chứa trong thư mục `skeleton` sẽ được sao chép vào thư mục chính. Cuối cùng, nhóm mới sẽ được tạo với cùng tên của tài khoản người dùng.

2. Bây giờ, bạn đã tạo tài khoản `dave`; người dùng này có thể đăng nhập vào hệ thống không?

Không, vì tài khoản `dave` đã bị khóa (xem dấu chấm than trong `/etc/shadow`).

```
# cat /etc/shadow | grep dave
dave:!:18015:0:99999:7:::
```

Nếu bạn đặt mật khẩu cho `dave`, tài khoản này sẽ được mở khóa. Bạn có thể thực hiện việc này bằng cách sử dụng lệnh `passwd`.

```
# passwd dave
Changing password for user dave.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

3. Hãy xác định ID người dùng (UID) và ID nhóm (GID) của `dave` và tất cả các thành viên của nhóm `dave`.

```
# cat /etc/passwd | grep dave
dave:x:1015:1019:~/home/dave:/bin/sh
# cat /etc/group | grep 1019
dave:x:1019:
```

UID và GID của `dave` lần lượt là 1015 và 1019 (trường thứ ba và thứ tư trong `/etc/passwd`) và nhóm `dave` không có thành viên nào (trường thứ tư trong `/etc/group` trống).

4. Hãy tạo các nhóm `sys_admin`, `web_admin` và `db_admin` và xác định ID nhóm (GID) của chúng.

```
# groupadd sys_admin
# groupadd web_admin
# groupadd db_admin
# cat /etc/group | grep admin
sys_admin:x:1020:
web_admin:x:1021:
db_admin:x:1022:
```

GID cho các nhóm `sys_admin`, `web_admin` và `db_admin` lần lượt là 1020, 1021 và 1022.

- Hãy thêm tài khoản người dùng mới có tên `carol` với UID 1035 và đặt `sys_admin` làm nhóm chính, `web_admin` và `db_admin` làm nhóm phụ.

```
# useradd -u 1035 -g 1020 -G web_admin,db_admin carol
# id carol
uid=1035(carol) gid=1020(sys_admin) groups=1020(sys_admin),1021(web_admin),1022(db_admin)
```

- Hãy xóa tài khoản người dùng `dave` và `carol` cũng như các nhóm `sys_admin`, `web_admin` và `db_admin` mà bạn đã tạo trước đó.

```
# userdel -r dave
# userdel -r carol
# groupdel sys_admin
# groupdel web_admin
# groupdel db_admin
```

- Hãy chạy lệnh `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` và mô tả đầu ra mà nó cung cấp cho bạn về quyền đối với tệp. Giả sử trong trường hợp hệ thống của bạn sử dụng mật khẩu ẩn, tệp nào trong số bốn tệp này bị ẩn vì lý do bảo mật?

```
# ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow
-rw-r--r-- 1 root root 853 mag 1 08:00 /etc/group
-rw-r----- 1 root shadow 1203 mag 1 08:00 /etc/gshadow
-rw-r--r-- 1 root root 1354 mag 1 08:00 /etc/passwd
-rw-r----- 1 root shadow 1563 mag 1 08:00 /etc/shadow
```

Các tệp `/etc/passwd` và `/etc/group` luôn có thể đọc được và bị ẩn vì lý do bảo mật. Khi mật khẩu bóng được sử dụng, bạn có thể thấy `x` trong trường thứ hai của các tệp này bởi vì mật khẩu được mã hóa cho người dùng và nhóm được lưu trữ trong `/etc/shadow` và

`/etc/gshadow`, và chúng chỉ có thể được đọc bởi người dùng gốc (trong một số hệ thống thì là bởi các thành viên thuộc nhóm `shadow`)

8. Hãy chạy lệnh `ls -l /usr/bin/passwd`. Bit đặc biệt nào được thiết lập và ý nghĩa của nó là gì?

```
# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 42096 mag 17 2015 /usr/bin/passwd
```

Lệnh `passwd` có bit SUID (ký tự thứ tư của dòng này) có nghĩa là lệnh được thực thi với các đặc quyền của chủ sở hữu tệp (quyền gốc). Đây là cách người dùng bình thường có thể thay đổi mật khẩu của họ.



**Linux
Professional
Institute**

5.3 Quản lý Quyền và Quyền sở hữu Tập

Tham khảo các mục tiêu LPI

[Linux Essentials version 1.6, Exam 010, Objective 5.3](#)

Khối lượng

2

Các lĩnh vực kiến thức chính

- Quyền và Quyền sở hữu Tập và Thư mục

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

- `ls -l`, `ls -a`
- `chmod`, `chown`



5.3 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	5 Bảo mật và Quyền trong Tệp
Mục tiêu:	5.3 Quản lý Quyền Truy cập và Quyền Sở hữu Tệp
Bài:	1 trên 1

Giới thiệu

Là một hệ thống đa người dùng, Linux cần có phương pháp để theo dõi xem ai sở hữu những tệp nào, và liệu người dùng có được phép thực hiện các thao tác trên tệp đó hay không. Điều này nhằm đảm bảo quyền riêng tư của những người dùng muốn giữ bí mật về nội dung tệp của họ, cũng như để đảm bảo sự cộng tác bằng cách cho phép nhiều người dùng có thể truy cập được một số tệp nhất định.

Điều này được thực hiện thông qua một hệ thống phân quyền ba cấp: mọi tệp trên đĩa đều được sở hữu bởi một người dùng và một nhóm người dùng và có ba bộ quyền: một cho chủ sở hữu của nó, một cho nhóm sở hữu tệp và một cho những người khác. Trong bài học này, ta sẽ học cách truy vấn quyền đối với một tệp và cách thao tác với chúng.

Truy vấn Thông tin về Tệp và Thư mục

Lệnh `ls` được sử dụng để lấy một danh sách nội dung của bất kỳ thư mục nào. Ở dạng cơ bản này, tất cả những gì bạn nhận được sẽ là tên của tệp:

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

Nhưng mỗi tệp đều sẽ có nhiều loại thông tin có sẵn về chúng bao gồm loại, kích thước, quyền sở hữu, v.v. Để xem các thông tin này, bạn phải yêu cầu `ls` cho ra một danh sách “dạng dài” 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 trong đầu ra ở trên đều mang một ý nghĩa riêng:

- Cột *đầu tiên* trong danh sách cho biết loại tệp và quyền.

Ví dụ: trên `drwxrwxr-x`:

- Ký tự đầu tiên (`d`) cho biết loại tệp.
- Ba ký tự tiếp theo (`rw`) cho biết quyền của chủ sở hữu tệp, còn được gọi là *người dùng* hoặc *u* (user).
- Ba ký tự tiếp theo (`rw`) cho biết quyền của *nhóm* sở hữu tệp, còn được gọi là *g* (group).
- Ba ký tự cuối cùng (`r-x`) cho biết quyền đối với bất kỳ ai khác, còn được gọi là *_* những người khác_ hoặc *o* (others).
- Cột *thứ hai* cho biết số lượng liên kết cứng hướng đến tệp đó. Đối với một thư mục, điều này có nghĩa là số lượng thư mục con cộng với một liên kết đến chính nó (`.`) và thư mục mẹ (`..`).
- Cột *thứ ba* và *thứ tư* tương ứng cho biết thông tin về quyền sở hữu của người dùng và nhóm sở hữu tệp.
- Cột *thứ năm* cho biết kích thước tệp tính bằng byte.
- Cột *thứ sáu* hiển thị ngày và giờ chính xác (tức *mốc thời gian*) khi tệp được sửa đổi lần cuối cùng.
- Cột *thứ bảy* và cột cuối cùng cho biết tên tệp.

Nếu bạn muốn xem kích thước của tệp khi ở định dạng “con người có thể đọc được”, hãy thêm tham số `-h` vào `ls`. Các tệp có kích thước nhỏ hơn một kilobyte sẽ hiển thị kích thước bằng byte. Các tệp lớn hơn một kilobyte và ít hơn một megabyte sẽ có thêm ký tự `K` đứng đằng sau kích thước

cho biết kích thước tính bằng kilobyte. Tương tự như vậy đối với kích thước tệp trong phạm vi megabyte (M) và gigabyte (G):

```
$ ls -lh
total 1,2G
drwxrwxr-x 2 carol carol 4,0K Dec 10 17:59 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
-rw----- 1 carol carol 528K Dec 10 10:43 picture.jpg
---xr-xr-x 1 carol carol  33 Dec 11 10:36 test.sh
-rwxr--r-- 1 carol carol 1,9K Dec 20 18:13 text.txt
-rw-rw-r-- 1 carol carol 2,6M Dec 11 22:14 Zipped.zip
```

Để chỉ hiển thị thông tin về một nhóm tệp cụ thể, hãy thêm tên của những tệp này vào `ls`:

```
$ ls -lh HugeFile.zip test.sh
total 1,2G
-rw-rw-r-- 1 carol carol 1,2G Dec 20 18:22 HugeFile.zip
---xr-xr-x 1 carol carol  33 Dec 11 10:36 test.sh
```

Còn Thư mục thì sao?

Nếu bạn thử truy vấn thông tin về một thư mục bằng cách sử dụng `ls -l`, nó sẽ hiển thị cho bạn một danh sách các nội dung của thư mục thay vào thông tin bạn cần:

```
$ 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, 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 Tập ẩn

Danh sách thư mục mà chúng ta đã truy xuất bằng cách sử dụng `ls -l` trước đó vẫn chưa được 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 đang bị ẩn. Trong Linux, các tệp có tên bắt đầu bằng dấu chấm (.) sẽ tự động bị ẩn. Để xem được 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` chỉ bị ẩn vì tên của nó bắt đầu bằng `.`

Tuy nhiên, các thư mục `.` và `..` là các thư mục đặc biệt. `.` là con trỏ tới thư mục hiện tại, trong khi `..` là 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 đặc biệt này.

TIP

Bạn có thể kết hợp nhiều tham số cho `ls` (và nhiều lệnh Linux khác). Ví dụ: `ls -l -a` có thể được viết là `ls -la`.

Tìm hiểu các loại Tệp

Chúng ta đã đề cập tới việc chữ cái đầu tiên trong mỗi đầu ra của `ls -l` sẽ mô tả loại tệp. Ba loại tệp phổ biến nhất là:

- (tệp thông thường)

Một tệp có thể chứa bất kỳ loại dữ liệu nào. 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 mềm)

“Tập” này là một con trỏ tới một tập hoặc thư mục khác ở một nơi nào đó khác trong hệ thống tệp.

Ngoài những loại trên, có ba loại tập khác mà ít nhất bạn nên biết tới (nhưng chúng nằm ngoài phạm vi của bài học này):

b (thiết bị khối)

Tập này là một thiết bị ảo hoặc một thiết bị vật lý, thường là đĩa hoặc các loại thiết bị lưu trữ khác. Ví dụ: ổ cứng đầu tiên trong hệ thống có thể được biểu thị bằng `/dev/sda`.

c (thiết bị ký tự)

Tập này là một thiết bị ảo hoặc vật lý. Cửa sổ dòng lệnh (như cửa sổ dòng lệnh chính trên `/dev/ttyS0`) và 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

Không được thay đổi bất kỳ quyền nào trên thiết bị khối, thiết bị ký tự hoặc ổ nối, trừ khi bạn biết rất rõ mình đang làm gì. Điều này có khiến hệ thống của bạn dừng hoạt động!

Hiểu về Quyền

Trong đầu ra của `ls -l`, các quyền của tệp được hiển thị ngay sau loại tệp dưới dạng ba nhóm, mỗi nhóm 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 nó đang bị thiếu một quyền cụ thể.

Quyền với Tệp

r

Viết tắt của *read* (đọc) và có giá trị bát phân là `4` (đừng lo, chúng ta sẽ thảo luận về hệ bát phân ngay sau đây). Nó đại diện cho quyền mở và đọc nội dung của một tệp.

w

Là viết tắt của *write* (ghi) và có giá trị bát phân là `2`. Nó đại diện cho quyền chỉnh sửa hoặc xóa tệp.

x

Là viết tắt của *execute* (thực thi) và có giá trị bát phân là `1`. Nó đại diện cho việc tệp có thể được

chạy dưới dạng tệp thực thi hoặc tệp lệnh.

Vì vậy, ví dụ nếu một tệp có quyền `rw-` có thể được đọc và ghi nhưng lại không thể được thực thi.

Quyền với Thư mục

r

Là viết tắt của *read* (đọc) và có giá trị bát phân là 4. Nó đại diện cho quyền đọc nội dung của thư mục, chẳng hạn như tên tệp. Tuy nhiên, nó *không* đại diện cho quyền đọc các tệp.

w

Là 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 hoặc thay đổi tên, quyền và chủ sở hữu của chúng. Nếu người dùng có quyền ghi trên một thư mục, người dùng đó có thể thay đổi quyền của bất kỳ tệp nào trong thư mục ngay cả khi người dùng không có quyền đối với tệp hoặc nếu tệp thuộc sở hữu của người dùng khác.

x

Là 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 (việc này cần quyền `r`).

Phần cuối cùng nghe có vẻ hơi khó hiểu. Ví dụ, hãy thử tưởng tượng rằng bạn có một thư mục có tên `Another_Directory` với các quyền sau:

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

Ngoài ra, hãy tưởng tượng rằng bên trong thư mục này, bạn có một tệp lệnh vỏ có tên `hello.sh` với các quyền sau:

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

Nếu bạn là người dùng `carol` và đang cố liệt kê nội dung của `Another_Directory`, bạn sẽ nhận được thông báo lỗi vì người dùng của bạn 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 họ có thể vào được thư mục. Do đó,

người dùng carol có thể truy cập vào các tệp bên trong thư mục, miễn là họ có quyền chính xác *đối với tệp tương ứng*. Trong ví dụ này, người dùng có toàn quyền đối với tệp lệnh hello.sh; vì vậy, họ *có thể* chạy tệp lệnh ngay cả khi họ *không* đọc được nội dung của thư mục chứa nó. Tất cả những gì họ cần ở đây là tên đầy đủ của tệp.

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

Như đã nói từ trước, các quyền được chỉ định theo trình tự: đầu tiên dành cho chủ sở hữu tệp, sau đó là dành cho nhóm sở hữu và cuối cùng là dành cho những người dùng khác. Bất cứ khi nào có một ai đó cố gắng thực hiện một hành động trên tệp, các quyền sẽ được kiểm tra theo cùng một cách. Trước tiên, hệ thống sẽ kiểm tra xem người dùng hiện tại có sở hữu tệp hay không, và nếu có thì hệ thống sẽ chỉ áp dụng nhóm quyền đầu tiên. Trường hợp ngược lại, nó sẽ kiểm tra xem người dùng hiện tại có thuộc nhóm sở hữu tệp hay không. Trong trường hợp có, 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 các quyền của chủ sở hữu mới có hiệu lực ngay cả khi quyền nhóm hoặc các quyền khác có hiệu lực cao hơn các quyền của chủ sở hữu.

Sửa đổi Quyền trong Tệp

Lệnh chmod được sử dụng để sửa đổi các quyền đối với một tệp và nó sẽ nhận 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 sẽ thực hiện thay đổi. Tuy nhiên, các quyền thay đổi có thể được mô tả theo hai chế độ (hay còn gọi là hai “mode”) khác nhau.

Chế độ đầu tiên được gọi là *chế độ tượng trưng*; nó cung cấp khả năng kiểm soát chi tiết và cho phép bạn thêm hoặc thu hồi một quyền duy nhất mà không cần sửa đổi các quyền khác trên nhóm. Chế độ thứ hai được gọi là *chế độ số*; nó dễ nhớ hơn và có thể được sử dụng nhanh hơn nếu bạn 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, ví dụ, các lệnh

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

và

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

sẽ tạo ra chính xác cùng một đầu ra - một tập có tập quyền như sau:

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

Bây giờ, hãy cùng tìm hiểu cách hoạt động của từng chế độ.

Chế độ Tượng trưng

Khi mô tả những quyền cần thay đổi trong chế độ *tượng trưng*, (các) ký tự đầu tiên cho biết (các) quyền của ai sẽ bị thay đổi: quyền của người dùng (u), của nhóm (g), của người khác (o) và/hoặc cho cả ba (a).

Sau đó, bạn cần cho lệnh biết phải làm gì: bạn có thể cấp quyền (+), thu hồi quyền (-) hoặc đặt quyền đó thành một giá trị cụ thể (=).

Cuối cùng, bạn sẽ chỉ định quyền nào bạn muốn tác động: đọc (r), ghi (w) hoặc thực thi (x).

Ví dụ: hãy tưởng tượng chúng ta có một tập có tên `text.txt` với tập quyền sau:

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

Nếu bạn muốn cấp quyền ghi cho các thành viên của nhóm sở hữu tập, bạn sẽ sử dụng tham số `g+w`. Sẽ dễ dàng hơn nếu bạn 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 kiểm tra kết quả với `ls`:

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

Nếu bạn muốn xóa quyền đọc đối với chủ sở hữu của cùng một tập, hãy nghĩ về nó như sau: “Đối với người dùng (u), thu hồi (-) quyền đọc (r)”. Vì vậy, ta có tham số là `u-r`:

```
$ chmod u-r text.txt
$ ls -l text.txt
```

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

Nếu chúng ta muốn đặt quyền chính xác là `rw-` cho tất cả mọi người thì sao? Nếu vậy, hãy nghĩ về nó như sau: “Cho tất cả (a), đặt chính xác quyền (=), đọc (r), ghi (w) và không thực thi (-)”. Vì thế nên ta có:

```
$ 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 có thể sửa đổi nhiều quyền cùng một lúc. Trong trường hợp này, hãy phân tách chúng bằng dấu phẩy (,):

```
$ chmod u+rw,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, viết 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` có chế độ đệ quy và nó sẽ hữu ích khi bạn muốn thay đổi quyền cho “tất cả các tệp bên trong một thư mục và các thư mục con của nó”. Để sử dụng nó, hãy thêm tham số `-R` sau tên lệnh và trước các quyền thay đổi, như thế này:

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

Lệnh này có thể được đọc là: “Đệ quy (-R), 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 nhiều tệp và thư mục con.

Chế độ Số

Trong *chế độ số*, các quyền sẽ được chỉ định theo một cách khác: dưới dạng giá trị số có ba chữ số trên ký hiệu bát phân, một hệ thống số cơ số 8.

Mỗi quyền có một giá trị tương ứng và chúng được chỉ định theo thứ tự sau: đầu tiên là đọc (r) được đại diện bằng 4, sau đó là ghi (w) được đại diện bằng 2 và cuối cùng là thực thi (x) được đại diện bằng 1. Nếu không có quyền, ta sẽ sử dụng giá trị không (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 số ba chữ số trên tập hợp quyền sẽ đại diện cho quyền của người dùng (u), chữ số thứ hai cho nhóm (g) và chữ số thứ ba cho những người khác (o). Nếu chúng ta muốn đặt quyền cho một tệp thành rw-rw---- thì 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
```

Bên cạnh đó, cú pháp trong *chế độ số* cũng giống như trong *chế độ tượng trưng*: tham số đầu tiên biểu thị các quyền bạn muốn thay đổi và tham số thứ hai trỏ tới tệp hoặc thư mục bạn sẽ thực hiện thay đổi.

TIP | Nếu giá trị quyền là *số lẻ*, tệp chắc chắn có thể thực thi được!

Vậy thì cú pháp sẽ như thế nào? *Chế độ số* được khuyên dùng nếu bạn muốn thay đổi quyền thành một giá trị cụ thể, ví dụ như 640 (rw- r-- ---).

Chế độ Tượng trưng sẽ hữu ích hơn khi bạn chỉ muốn lật một giá trị cụ thể, bất kể các quyền hiện tại đối với tệp là như thế nào. 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 hoặc thậm chí chạm đến 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 nó khá là đơn giản:

```
chown username:groupname filename
```

Ví dụ: hãy thử kiểm tra 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, hãy sửa đổi nhóm sở hữu tệp

thành một số nhóm khác, chẳng hạn như `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 thuộc nhóm sở hữu tệp. Trong ví dụ trên, người dùng `carol` không cần phải là thành viên của nhóm `students`. Tuy nhiên, người dùng này phải là thành viên của nhóm để chuyển quyền sở hữu nhóm của tệp cho nhóm đó.

Người dùng hoặc nhóm có thể được bỏ qua nếu bạn không muốn thay đổi chúng. Vì vậy, để chỉ thay đổi nhóm sở hữu tệp, hãy sử dụng `chown :students text.txt`. Để chỉ thay đổi người dùng, lệnh sẽ là `chown carol text.txt`. Ngoài ra, bạn có thể sử dụng lệnh `chgrp students text.txt` để chỉ thay đổi nhóm.

Trừ khi bạn là quản trị viên hệ thống (root), bạn sẽ không thể thay đổi quyền sở hữu tệp do người dùng khác hoặc nhóm mà bạn không thuộc về. Nếu bạn cố làm điều này, bạn sẽ nhận được thông báo lỗi `Operation not permitted` (Thao tác không được phép).

Nhóm Truy vấn

Trước khi thay đổi quyền sở hữu tệp, việc biết rõ nhóm nào tồn tại trên hệ thống, người dùng nào là thành viên của nhóm và người dùng thuộc nhóm nào sẽ là khá hữu ích. Bạn có thể hoàn thành các tác vụ đó bằng hai lệnh `groups` và `groupmems`.

Để xem những nhóm nào tồn tại trên hệ thống của bạn, chỉ cần gõ `groups`:

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

Và nếu bạn muốn biết người dùng thuộc nhóm nào, hãy thêm tên người dùng làm tham số:

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

Để làm ngược lại và hiển thị những người dùng nào thuộc về một nhóm, hãy sử dụng `groupmems`. Tham số `-g` sẽ chỉ định nhóm và `-l` sẽ liệt kê tất cả các thành viên của nhóm:

```
$ sudo groupmems -g cdrom -l
```

```
carol
```

TIP

`groupmems` chỉ có thể được chạy với quyền gốc, tức quản trị viên hệ thống. Nếu bạn hiện chưa đăng nhập với quyền gốc, hãy thêm `sudo` trước lệnh.

Quyền Đặc biệt

Bên cạnh quyền đọc, ghi và thực thi đối với 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 ký hiệu hoặc số:

Bit dính

Bit dính (hay còn được gọi là *cờ xóa bị hạn chế*) có giá trị bát phân là 1 và khi ở chế độ tượng trưng sẽ được biểu thị bằng `t` trong các quyền của những người khác. Điều này chỉ áp dụng cho các thư mục và trên Linux, nó sẽ ngăn người dùng xóa hoặc đổi tên tệp trong 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` trên các quyền dành cho *những người khác* trong 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 số, các quyền đặc biệt được chỉ định bằng cách sử dụng “ký hiệu 4 chữ số” với chữ số đầu tiên biểu thị quyền đặc biệt để thực hiện theo. Ví dụ: để đặt bit dính (giá trị 1) cho thư mục `Another_Directory` ở chế độ số, 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
```

Thiết lập GID

Thiết lập GID (hay còn được gọi là bit SGID hoặc bit Đặt ID nhóm) có giá trị bát phân 2 và khi ở chế độ tượng trưng sẽ được biểu thị bằng `s` trong các quyền *nhóm*. Điều này có thể được áp dụng cho các tệp hoặc thư mục thực thi. Trên các tệp thực thi, nó sẽ cấp cho quá trình xuất phát từ việc thực thi quyền truy cập tệp đến các đặc quyền của nhóm sở hữu tệp. Khi được áp dụng cho các thư mục, nó sẽ làm cho mọi tệp hoặc thư mục được tạo bên dưới nó 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 đối với *nhóm* trong đầu ra của `ls -l`:

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

Để thêm quyền SGID vào tệp ở chế độ tương 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 bạn hiểu rõ hơn về tác dụng của SGID đối với một thư mục. Giả sử chúng ta có một thư mục tên là `Sample_Directory` thuộc sở hữu của 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 chuyển sang thư mục này và tạo một tệp trống bên trong nó bằng cách sử dụng lệnh `touch`. 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 được sở hữu bởi người dùng `carol` và nhóm `carol`. Tuy nhiên, nếu thư mục có tập quyền SGID, kết quả sẽ khác. Đầu tiên, hãy 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 các quyền của nhóm cho biết rằng bit SGID đã được đặt. Bây giờ, hãy chuyển sang 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
```

Như chúng ta có thể thấy, 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, tức `users`.

Đặt UID

SUID (hay còn gọi là Đặt ID người dùng) có giá trị bát phân 4 và được biểu thị bằng ký tự `s` trên quyền *người dùng* ở chế độ tượng trưng. Nó chỉ áp dụng cho các tệp và hành vi của nó cũng tương tự như bit SGID; tuy nhiên, quy trình sẽ chạy với các đặc quyền của *người dùng* sở hữu tệp. Các tệp có bit SUID sẽ hiển thị `s` thay thế `x` trên các quyền của người dùng trong đầu ra của `ls -l`:

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

Bạn có thể kết hợp nhiều quyền đặc biệt trong một tham số bằng cách cộng chúng lại với nhau. Vì vậy, để đặt SGID (giá trị 2) và SUID (giá trị 4) ở chế độ số cho tệp lệnh `test.sh` với quyền 755, ta sẽ nhập:

```
$ 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 (hầu hết các cửa sổ dòng lệnh ngày nay đều có hỗ trợ), bạn có thể nhanh chóng xem liệu các quyền đặc biệt này có được đặt hay không bằng cách nhìn 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 màu xanh lam. Điều tương tự cũng áp dụng cho các tệp có bit SGID (nền vàng) và SUID (nền đỏ). Màu sắc có thể khác nhau 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 cách sử dụng lệnh `mkdir emptydir`. Bây giờ, bằng cách sử dụng `ls`, hãy liệt kê các quyền đối với thư mục `emptydir`.

2. Hãy tạo một tệp trống có tên `emptyfile` bằng lệnh `touch emptyfile`. Bây giờ, hãy sử dụng `chmod` với ký hiệu tượng trưng và thêm quyền thực thi cho chủ sở hữu của tệp `emptyfile`, đồng thời xóa quyền ghi và thực thi đối với những người khác. Hãy thực hiện việc này chỉ bằng một lệnh `chmod`.

3. Các quyền đối với tệp có tên `text.txt` sau khi bạn sử dụng lệnh `chmod 754 text.txt` sẽ là gì?

4. Giả sử một tệp có tên `test.sh` là một tệp lệnh vỏ với các quyền và quyền sở hữu sau:

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

- Chủ sở hữu của tệp có những quyền gì?

- Nếu người dùng `john` chạy tệp lệnh này, tệp lệnh sẽ được chạy theo đặc quyền của người dùng nào?

- Sử dụng ký hiệu số, cú pháp nào của `chmod` có thể “huỷ 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
```

Loại tệp nào là `sdb1`? Và ai có thể ghi trên 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 cách sử dụng ký hiệu số gồm 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 `touch emptyfile`. Bây giờ, hãy “xoá hết” quyền đối với tệp có `chmod 000 emptyfile`. Điều gì sẽ xảy ra nếu bạn thay đổi quyền cho `emptyfile` bằng cách chỉ chuyển *một* giá trị cho `chmod` ở chế độ số, chẳng hạn như `chmod 4 emptyfile`? Nếu chúng ta sử dụng hai giá trị, chẳng hạn như `chmod 44 emptyfile`, thì sẽ thế nào? Chúng ta có thể học được gì về cách `chmod` đọc giá trị số?

2. Bạn có thể thực thi một tệp mà bạn đã thực thi nhưng không có quyền đọc (`--x`) không? Tại sao?

3. Hãy xem các quyền đối với thư mục tạm thời `/tmp` trên hệ thống Linux:

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

Người dùng, nhóm và những người khác có đầy đủ quyền. Nhưng một người dùng thông thường có thể xóa *bất kỳ* một tệp nào trong thư mục này không? Tại sao?

4. Tệp có tên `test.sh` có các quyền sau: `-rwsr-xr-x`, nghĩa là bit SUID đã được đặt. 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?

5. Làm cách nào để có thể tạo một thư mục có tên `Box` mà trong đó tất cả các tệp sẽ được sở hữu tự động bởi nhóm `users` và chỉ người dùng đã tạo ra chúng mới có thể xóa được?

Tóm tắt

Là một hệ thống đa người dùng, Linux cần có một phương pháp nhất định để theo dõi xem ai sở hữu và ai có thể truy cập vào từng tệp. Điều này được thực hiện thông qua hệ thống phân quyền ba cấp và trong bài học này, chúng ta đã tìm hiểu về tất cả các cách thức hoạt động của hệ thống này.

Trong bài học này, bạn đã học cách sử dụng `ls` để nhận thông tin về quyền của tệp, cách kiểm soát hoặc thay đổi người có thể tạo, xóa hoặc sửa đổi tệp bằng `chmod` (cả bằng ký hiệu số và *tượng trưng*) và cách thay đổi quyền sở hữu của các tệp bằng `chown` và `chgrp`.

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 hoặc không.

`chmod`

Thay đổi quyền của một tệp hoặc thư mục.

`chown`

Thay đổi người dùng và/hoặc nhóm sở hữu của tệp hoặc thư mục.

`chgrp`

Thay đổi nhóm sở hữu của một tệp hoặc thư mục.

Đáp án Bài tập Hướng dẫn

1. Hãy tạo một thư mục có tên `emptydir` bằng cách sử dụng lệnh `mkdir emptydir`. Bây giờ, bằng cách sử dụng `ls`, hãy liệt kê các quyền đối với thư mục `emptydir`.

Thêm tham số `-d` vào `ls` để xem thuộc tính tệp của thư mục thay vì liệt kê nội dung của thư mục. Vì vậy, câu trả lời sẽ là:

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

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, hãy sử dụng `chmod` với ký hiệu tượng trưng và thêm quyền thực thi cho chủ sở hữu của tệp `emptyfile`, đồng thời xóa quyền ghi và thực thi đối với những người khác. Hãy thực hiện việc này chỉ bằng một lệnh `chmod`.

Hãy nghĩ về nó theo cách này:

- “Đối với người dùng sở hữu tệp (`u`), thêm (+) quyền thực thi (`x`)”, vậy nên sẽ là `u+x`.
- “Đối với nhóm (`g`) và những người dùng khác (`o`), xóa (-) quyền ghi (`w`) và thực thi (`x`)”, vậy nên sẽ là `go-wx`.

Để kết hợp hai bộ quyền này, ta thêm dấu phẩy giữa chúng. Vì vậy, kết quả cuối cùng sẽ là:

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

3. Các quyền đối với tệp có tên `text.txt` sau khi bạn sử dụng lệnh `chmod 754 text.txt` sẽ là gì?

Hãy nhớ rằng trong ký hiệu số, mỗi chữ số sẽ đại diện cho một bộ ba quyền, mỗi quyền có một giá trị tương ứng: *đọc* là 4, *ghi* là 2, *thực thi* là 1 và không có quyền là 0. Chúng ta nhận giá trị cho một chữ số bằng cách thêm các giá trị tương ứng cho mỗi quyền. 7 là 4+2+1 hoặc `rxw`, 5 là 4+0+1, vì vậy `r-x` và 4 sẽ là chỉ được đọc, hoặc `r--`. Các quyền đối với `text.txt` sẽ là

```
rxw-r-x---
```

4. Giả sử một tệp có tên `test.sh` là một tệp lệnh với các quyền và quyền sở hữu sau:

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

- Chủ sở hữu của tệp có những quyền gì?

Quyền của 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”.

- Nếu người dùng `john` chạy tệp lệnh này, tệp lệnh sẽ được chạy theo đặc quyền của người dùng nào?

Hãy chú ý đến các quyền cho *nhóm*. Chúng là `r-s`, có nghĩa là bit SGID đã được đặt. Nhóm sở hữu tệp này là `root`; do đó, tệp lệnh (ngay cả khi được khởi chạy bởi người dùng thông thường) sẽ được chạy với quyền gốc.

- Sử dụng ký hiệu số, cú pháp nào của `chmod` có thể “huỷ thiết lập” quyền đặc biệt được cấp cho tệp này?

Chúng ta có thể “huỷ thiết lập” các quyền đặc biệt bằng cách chuyển 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
```

Loại tệp nào là `sdb1`? Và ai có thể ghi trên nó?

Ký tự đầu tiên của đầu ra từ `ls -l` cho biết loại tệp. `b` là *thiết bị khối*, thường là đĩa (disk - bên trong hoặc bên ngoài), được kết nối với máy. Chủ sở hữu (`root`) và bất kỳ người dùng nào trong nhóm `disk` đều 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 cách sử dụng ký hiệu số gồm 4 chữ số.

Các quyền tương ứng như sau (trong ký hiệu số):

Another_Directory

Trả lời: 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

Trả lời: 0044

Không có quyền đặc biệt (vì vậy nên chữ số đầu tiên là 0), không có quyền cho người dùng (---) và quyền chỉ đọc (r-- --) cho nhóm và những người khác.

HugeFile.zip

Trả lời: 0664

Không có quyền đặc biệt, vì vậy nên chữ số đầu tiên là 0. 6 (rw-) cho người dùng và nhóm, 4 (r-- --) cho những người khác.

Sample_Directory

Trả lời: 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

- Hãy thử điều này trên cửa sổ dòng lệnh: tạo một tệp trống có tên `emptyfile` bằng lệnh `touch emptyfile`. Bây giờ, hãy “xoá hết” quyền đối với tệp có `chmod 000 emptyfile`. Điều gì sẽ xảy ra nếu bạn thay đổi quyền cho `emptyfile` bằng cách chỉ chuyển *một* giá trị cho `chmod` ở chế độ số, chẳng hạn như `chmod 4 emptyfile`? Nếu chúng ta sử dụng hai giá trị, chẳng hạn như `chmod 44 emptyfile`, thì sẽ thế nào? Chúng ta có thể học được gì về cách `chmod` đọc giá trị số?

Hãy nhớ rằng chúng ta đã “xoá hết” các quyền đối với “tệp trống”. Vì vậy, trạng thái ban đầu của nó sẽ là:

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

Bây giờ, 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
```

Các quyền cho *những người khác* đã được thay đổi. Và nếu chúng ta thử hai giá trị, chẳng hạn như `chmod 44 emptyfile`, thì sẽ thế nào?

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

Bây giờ, các quyền dành cho *nhóm* và *những người khác* đã bị ảnh hưởng. Từ đó, chúng ta có thể kết luận rằng trong ký hiệu số, `chmod` sẽ đọc giá trị “ngược” từ chữ số ít quan trọng nhất (*những người khác*) đến chữ số quan trọng nhất (*người dùng*). Nếu bạn đi qua 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 sửa đổi quyền cho *người dùng*, *nhóm*, *những người khác* và các quyền đặc biệt.

- Bạn có thể thực thi một tệp mà bạn đã thực thi nhưng không có quyền đọc (`--x`) không? Tại sao?

Ban đầu, câu trả lời có vẻ là khá hiển nhiên: Nếu bạn có quyền thực thi, tệp sẽ chạy. Điều này áp dụng cho các chương trình ở định dạng nhị phân được thực thi trực tiếp bởi nhạ hân. Tuy

nhiên, có những chương trình (ví dụ: tệp lệnh vỏ) trước tiên phải được đọc và diễn giải; do đó, trong những trường hợp này, quyền đọc (r) cũng phải được đặt.

3. Hãy xem các quyền đối với thư mục tạm thời /tmp trên hệ thống Linux:

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

Người dùng, nhóm và những người khác có đầy đủ quyền. Nhưng một người dùng thông thường có thể xóa *bất kỳ* một tệp nào trong thư mục này không? Tại sao?

/tmp là thư mục *luôn ghi được*, nghĩa là bất kỳ người dùng nào cũng có thể ghi vào thư mục đó. Nhưng chúng ta không muốn một người dùng nào đó sửa đổi các tệp do một người khác tạo ra; vì vậy, *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 tệp trong /tmp nhưng chỉ trong trường hợp họ là người đã tạo tệp đó.

4. Tệp có tên test.sh có các quyền sau: -rwsr-xr-x, nghĩa là bit SUID đã được đặt. 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) thay thế cho x trong đầu ra của ls -l; vì vậy, hệ thống cần một cách để hiển thị xem người dùng có quyền thực thi hay không. Điều này được thực hiện bằng cách thay đổi kiểu định dạng (chữ hoa hoặc chữ thường) của ký tự đặc biệt.

Chữ thường s trên nhóm quyền đầu tiên có nghĩa là người dùng sở hữu tệp có quyền thực thi và bit SUID đã được đặt. Chữ hoa S có nghĩa là người dùng sở hữu tệp thiếu (-) quyền thực thi và bit SUID đã được đặt.

Điều tương tự cũng có thể xảy ra với SGID. Chữ thường s trên nhóm quyền thứ hai có nghĩa là nhóm sở hữu tệp có quyền thực thi và bit SGID đã được đặt. Chữ hoa S có nghĩa là nhóm sở hữu tệp thiếu (-) quyền thực thi và bit SGID đã được đặt.

Điều này cũng đúng với bit dính được biểu thị bằng t trong nhóm quyền thứ ba. Chữ thường t có nghĩa là bit dính đã được đặt và những những người khác có quyền thực thi. Chữ hoa T có

nghĩa là bit dính đã được đặt và những người khác không có quyền thực thi.

5. Làm cách nào để có thể tạo một thư mục có tên `Box` mà trong đó tất cả các tệp sẽ được sở hữu tự động bởi nhóm `users` và chỉ người dùng đã tạo ra chúng mới có thể xóa được?

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 tự động được gán cho nhóm `users`. Chúng ta có thể làm điều này bằng cách đặt nhóm này làm chủ sở hữu của thư mục, sau đó đặt 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 chúng ta không quan tâm đến các quyền khác là gì và chỉ muốn “lật” các bit đặc biệt nên việc sử dụng chế độ tượng trưng là điều hợp lý:

```
$ chown :users Box/
$ chmod g+wx 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 phần cuối cùng: đảm bảo rằng chỉ người dùng đã tạo tệp mới được phép xóa tệp đó. Điều này được thực hiện bằng cách đặt bit dính (được biểu thị bằng `t`) trên thư mục. Hãy nhớ rằng nó được đặt trên quyền của những người khác (`o`).

```
$ chmod o+t Box/
```

Các quyền trên thư mục `Box` sẽ xuất hiện 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+wx,o+t Box/
```

Điểm cộng nếu bạn nghĩ tới điều này.



5.4 Các Thư mục và Tập đặc biệt

Tham khảo các mục tiêu LPI

[Linux Essentials v1.6, Exam 010, Objective 5.4](#)

Khối lượng

1

Các lĩnh vực kiến thức chính

- Sử dụng các tập và thư mục tạm thời
- Liên kết tượng trưng

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

- `/tmp/`, `/var/tmp/` và Sticky Bit
- `ls -d`
- `ln -s`



5.4 Bài 1

Chứng chỉ:	Linux Essentials
Phiên bản:	1.6
Chủ đề:	5 Bảo mật và Quyền trong Tập
Mục tiêu:	5.4 Thư mục và Tập đặc biệt
Bài:	1 trên 1

Giới thiệu

Trong Linux, mọi thứ đều được coi là một tập. Tuy nhiên, một số tập sẽ được xử lý đặc biệt do vị trí lưu trữ của chúng (chẳng hạn như tập tạm thời) hoặc cách chúng tương tác với hệ thống tập (chẳng hạn như các liên kết). Trong bài học này, chúng ta sẽ tìm hiểu về vị trí của các tập như vậy, cách chúng hoạt động và cách ta quản lý chúng.

Tập Tạm thời

Các tập tạm thời là các tập được các 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 quy trình đang chạy, nhật ký sự cố, tập scratch lưu tự động, tập trung gian được sử dụng trong quá trình chuyển đổi tập, tập bộ đệm, v.v.

Vị trí của các Tập tạm thời

Phiên bản 3.0 của *Tiêu chuẩn Phân cấp Hệ thống Tập* (FHS) xác định các vị trí tiêu chuẩn cho các tập tạm thời trong 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 cho rằng các tệp được ghi ở đây sẽ được giữ nguyên giữa các lần gọi chương trình. *Khuyến nghị* ở đây là thư mục này nên được xóa (xóa tất cả các tệp) trong khi khởi động hệ thống (điều này là *không bắt buộc*).

/var/tmp

Một vị trí khác dành cho các tệp tạm thời nhưng nó *sẽ không bị xóa* trong quá trình khởi động hệ thống - tức là các tệp được lưu trữ ở đây thường sẽ tồn tại giữa các lần khởi động lại.

/run

Thư mục này chứa dữ liệu của biến thời gian chạy được sử dụng bởi các quy trình đang chạy, chẳng hạn như các tệp định danh quy trình (.pid). Các chương trình cần nhiều hơn một tệp thời gian chạy có thể sẽ tạo các thư mục con tại đây. Vị trí này *phải được xóa sạch* trong khi khởi động hệ thống. Mục đích của thư mục này đã từng được thực hiện bởi /var/run và trên một số hệ thống /var/run có thể là một liên kết tượng trưng tới /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 chúng ta vẫn nên tuân thủ các quy ước do FHS đặt ra.

Quyền đối với các Tệp tạm thời

Việc có các thư mục tạm thời toàn hệ thống trên một hệ thống đa người dùng gây ra một số thách thức liên quan đến quyền truy cập. Lúc đầu, người ta có thể nghĩ rằng những thư mục như vậy sẽ “luôn có thể ghi được” - tức là bất kỳ người dùng nào cũng có thể ghi hoặc xóa dữ liệu trong đó. Nhưng nếu điều này là đúng, làm cách nào để chúng ta có thể ngăn người dùng xóa hoặc sửa đổi tệp do người khác tạo ra?

Giải pháp ở đây là một quyền đặc biệt được gọi là *bit dính* áp dụng cho cả thư mục và tệp. Tuy nhiên, vì lý do bảo mật, nhân Linux sẽ bỏ qua bit dính khi nó được áp dụng cho các tệp. Khi bit đặc biệt này được đặt cho một thư mục, nó sẽ ngăn người dùng xóa hoặc đổi tên tệp trong thư mục đó trừ khi họ sở hữu tệp đó.

Các thư mục có bit dính sẽ hiển thị t thay thế x trên quyền đối với *những người khác* trong đầu ra của `ls -l`. Ví dụ: hãy kiểm tra quyền đối với thư mục /tmp và /var/tmp:

```
$ ls -ldh /tmp/ /var/tmp/
drwxrwxrwt 25 root root 4,0K Jun  7 18:52 /tmp/
drwxrwxrwt 16 root root 4,0K Jun  7 09:15 /var/tmp/
```

Như có thể thấy, bằng cách thay thế t bằng x trên quyền đối với *những người khác*, cả hai thư mục

đều đã có bit dính.

Để đặt bit dính trên một thư mục bằng cách sử dụng `chmod` ở chế độ số, hãy sử dụng ký hiệu bốn chữ số và lấy 1 làm chữ số đầu tiên. Ví dụ:

```
$ chmod 1755 temp
```

sẽ đặt bit dính cho thư mục có tên `temp` và các quyền là `rxr-xr-t`.

Khi sử dụng chế độ ký hiệu, hãy sử dụng tham số `t`. Vậy nên `+t` là để đặt bit dính và `-t` sẽ huỷ thiết lập của nó. Vì thế nên ta có:

```
$ chmod +t temp
```

Hiểu về Liên kết

Chúng ta đã nói qua rằng trong Linux, mọi thứ đều được coi là một tập. Nhưng có một loại tập *đặc biệt* được gọi là *liên kết* (link), và trên hệ thống Linux ta có hai loại liên kết:

Liên kết tượng trưng

Hay còn được gọi là *liên kết mềm* (soft link) trỏ đến đường dẫn của một tập khác. Nếu bạn xóa tập mà liên kết trỏ tới (hay còn được gọi là *đích*) thì liên kết sẽ vẫn tồn tại nhưng nó sẽ “ngừng hoạt động” vì nó hiện đang không trỏ tới bất cứ thứ gì.

Liên kết cứng

Hãy coi liên kết cứng (hard link) là tên thứ hai cho tập gốc. Chúng *không* phải là các bản sao mà thay vào đó là một mục nhập bổ sung trong hệ thống tập trỏ đến cùng một vị trí (inode) trên đĩa.

TIP

inode là một cấu trúc dữ liệu lưu trữ các thuộc tính cho một đối tượng (ví dụ như tập hoặc thư mục) trên một hệ thống tập. Trong số các thuộc tính đó sẽ có tên tập, quyền, quyền sở hữu và khối đĩa mà dữ liệu về đối tượng được lưu trữ. Hãy coi nó như một mục nhập trong một chỉ mục (index) - cũng chính vì thế mà nó được gọi là “nút chỉ mục” (index node).

Làm việc với Liên kết Cứng

Tạo Liên kết Cứng

Lệnh để tạo liên kết cứng trên Linux là `ln`. Cú pháp cơ bản của nó là:


```
$ ln TARGET LINK_NAME
```

TARGET (ĐÍCH) phải tồn tại (đây là tệp mà liên kết sẽ trở đến) và nếu đích không có trong thư mục hiện tại hoặc nếu bạn muốn tạo liên kết ở nơi khác, bạn *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 bạn bỏ qua tham số cuối cùng (LINK_NAME), một liên kết có cùng tên với đích 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 trở đến cùng một dữ liệu trên đĩa. Tất cả những cái tên này đều ngang hàng với nhau và có thể được sử dụng để chỉ một tệp. Nếu bạn thay đổi nội dung của một trong những cái tên này, nội dung của tất cả các tên khác trở đến tệp đó sẽ thay đổi vì tất cả các tên này đều trở đến cùng một dữ liệu. Nếu bạn xóa một trong số chúng, các tên khác sẽ vẫn hoạt động.

Điều này xảy ra là bởi khi bạn “xóa” một tệp, dữ liệu sẽ không thực sự bị xóa khỏi đĩa. Hệ thống chỉ cần xóa mục trên bảng hệ thống tệp trở đến inode tương ứng với dữ liệu trên đĩa. Nhưng nếu bạn có một mục thứ hai trở đến cùng một inode, bạn vẫn có thể truy cập dữ liệu đó. Hãy coi nó như hai con đường hội tụ về một điểm. Ngay cả khi bạn chặn hoặc chuyển hướng một trong hai con đường, bạn vẫn có thể đến đích bằng con đường còn lại.

Bạn có thể kiểm tra điều này bằng cách sử dụng tham số `-li` của `ls`. Hãy xem xét các nội dung sau 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 là số inode. Bạn có thấy rằng cả tệp `hardlink` và tệp `target.txt` đều có cùng một số (3806696) không? Điều này xảy ra do tệp này là liên kết cứng của tệp kia.

Nhưng tệp nào là bản gốc và tệp nào là liên kết? Thực sự thì chúng ta không thể biết được bởi dù

với bất kỳ một mục đích thực tế nào thì cả hai tập cũng đều có chức năng giống nhau.

Hãy lưu ý rằng mọi liên kết cứng trỏ đến một tập sẽ tăng *số lượng liên kết* của tập. Đây là con số đứng ngay sau các quyền trong đầu ra của `ls -l`. Theo mặc định, mọi tập đều có số lượng liên kết là 1 (các thư mục có số lượng là 2) và mọi liên kết cứng tới nó sẽ tăng số lượng lên một. Đó là lý do số lượng liên kết là 2 trên các tập trong danh sách ở trên.

Ngược lại với các liên kết tượng trưng, bạn chỉ có thể tạo các liên kết cứng tới các tập, và cả liên kết và đích phải nằm trong cùng một hệ thống tập.

Di chuyển và loại bỏ các Liên kết Cứng

Vì các liên kết cứng cũng được coi là các tập thông thường nên chúng có thể bị xóa bằng `rm` và có thể bị đổi tên hoặc di chuyển xung quanh hệ thống tập bằng `mv`. Và vì một liên kết cứng trỏ đến cùng một inode của 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 tập có tên là `softlink` trong thư mục `/home/carol/Documents/` trỏ tới tập `target.txt` trong thư mục hiện tại.

Cũng giống như liên kết cứng, bạn 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 tập. Bạn có thể tạo các liên kết mềm đến các tập 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, bạn nên lưu ý rằng trừ khi một đường dẫn được chỉ định đầy đủ, vị trí của đích sẽ được hiểu theo *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. 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 cùng 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 bạn di chuyển liên kết đến đâu thì nó vẫn sẽ hoạt động vì nó trở 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. Hãy tưởng tượng một chương trình cần tạo một tệp tạm thời để sử dụng một lần và sẽ không bao giờ cần tới nó nữa sau khi đóng chương trình. Thư mục chính xác để tạo tệp này là thư mục nào?

2. Thư mục tạm thời *phải* bị xóa trong quá trình khởi động là thư mục nào?

3. Tham số nào của `chmod` trong chế độ *tượng trưng* có thể kích hoạt bit dính trên một thư mục?

4. Hãy tưởng tượng có một tệp có tên là `document.txt` trong thư mục `/home/carol/Documents`. Lệnh dùng để tạo liên kết tượng trưng đến nó và có tên là `text.txt` trogn thư mục hiện tại là gì?

5. Hãy giải thích sự khác biệt giữa một liên kết cứng tới một tệp và một bản sao của tệp đó.

Bài tập Mở rộng

- 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 trở đến cùng một inode giống như `recipes.txt`. Điều gì sẽ xảy ra với liên kết mềm `rezepte.txt` nếu như tên `receitas.txt` bị xóa? Tại sao?

- Hãy tưởng tượng bạn có một ổ đĩa flash được cắm vào hệ thống của mình và được gắn 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 gõ lệnh:

```
$ ln /media/youruser/FlashA/esquema.pdf ~/schematics.pdf
```

Chuyện gì sẽ xảy ra? Tại sao?

- 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 chuyển tham số nào cho `ls` để xem mỗi tệp ở trong inode nào?

4. Hãy tưởng tượng trong thư mục `~/Documents` của bạn có một tệp tên là `clients.txt` chứa một số tên khách hàng 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
```

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, bạn đã học về:

- Nơi lưu trữ các tệp tạm thời.
- Quyền đặc biệt áp dụng cho chúng.
- 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 xóa chúng.

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

- `ln`
- Tham số `-i` thành `ls`

Đáp án Bài tập Hướng dẫn

1. Hãy tưởng tượng một chương trình cần tạo một tệp tạm thời để sử dụng một lần và sẽ không bao giờ cần tới nó nữa sau khi đóng chương trình. Thư mục chính xác để tạo tệp này là thư mục nào?

Vì chúng ta không quan tâm đến tệp sau khi chương trình chạy xong nên thư mục chính xác sẽ là `/tmp`.

2. Thư mục tạm thời *phải* bị xóa trong quá trình khởi động là thư mục nào?

Thư mục là `/run` hoặc trên một số hệ thống là `/var/run`.

3. Tham số nào của `chmod` trong chế độ *tượng trưng* có thể kích hoạt bit dính trên một thư mục?

Biểu tượng cho bit dính trong chế độ tượng trưng là `t`. Vì chúng ta muốn kích hoạt (thêm) quyền này vào thư mục nên tham số phải là `+t`.

4. Hãy tưởng tượng có một tệp có tên là `document.txt` trong thư mục `/home/carol/Documents`. Lệnh dùng để tạo liên kết tượng trưng đến nó và có tên là `text.txt` trogn thư mục hiện tại là gì?

`ln -s` là lệnh tạo liên kết tượng trưng. Vì bạn phải chỉ định đường dẫn đầy đủ đến tệp mà bạn đang liên kết tới, nên lệnh sẽ là:

```
$ ln -s /home/carol/Documents/document.txt text.txt
```

5. Hãy giải thích sự khác biệt giữa một liên kết cứng tới một tệp và một bản sao của tệp đó.

Liên kết cứng chỉ là một tên gọi khác cho tệp. Mặc dù nó trông giống như một bản sao của tệp gốc nhưng đối với mọi mục đích, 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 đối với nội dung của liên kết sẽ được phản ánh trên bản gốc và ngược lại. Bản sao là một thực thể hoàn toàn độc lập, chiếm một vị trí khác trên đĩa. Các thay đổi đối với bản sao sẽ không được phản ánh trên bản gốc và ngược lại.

Đáp án Bài tập Mở rộng

- 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 trở đến cùng một inode 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 trở đến tên chứ không phải inode, và tên `receitas.txt` không còn tồn tại ngay cả khi dữ liệu vẫn còn trên đĩa dưới tên `recipes.txt`.

- Hãy tưởng tượng bạn có một ổ đĩa flash được cắm vào hệ thống của mình và được gắn trên `/media/youruser/FlashA`. Bạn muốn tạo trong thư mục chính của mình một liên kết có tên là `schematics.pdf` và trở tới tệp `esquema.pdf` trong thư mục gốc của ổ đĩa flash. Vì vậy, bạn 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 thiết bị chéo không hợp lệ) và nó đã làm rõ lý do: các liên kết cứng không thể trở 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 trở 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 trở 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 chuyển tham số nào cho `ls` để xem mỗi tệp ở trong inode nào?

Tham số là `-li`. Inode 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` chứa một số tên khách hàng 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
```

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 “khó”; 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 có liên quan đến 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.

Ấn bản

© 2024 bởi Linux Professional Institute: Tài liệu Học tập, “Linux Essentials (Version 1.6)”.

PDF được tạo vào: 2024-04-29

Ấn phẩm này được cấp phép theo Giấy phép Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0). Để xem bản sao của giấy phép này, hãy truy cập

<https://creativecommons.org/licenses/by-nc-nd/4.0/>

Mặc dù Linux Professional Institute đã rất nỗ lực một cách thiện chí để đảm bảo rằng các thông tin và hướng dẫn trong các tài liệu này là chính xác, Linux Professional Institute từ chối mọi trách nhiệm đối với các lỗi hoặc thiếu sót, bao gồm nhưng không giới hạn trách nhiệm đối với thiệt hại do việc sử dụng hoặc phụ thuộc vào tài liệu này. Việc sử dụng các thông tin và hướng dẫn có trong tài liệu này là rủi ro của riêng người dùng. Nếu bất kỳ mẫu mã hoặc công nghệ nào khác mà tài liệu này nhắc tới hoặc mô tả cần tuân theo giấy phép mã nguồn mở hoặc quyền sở hữu trí tuệ của người khác, người dùng có trách nhiệm đảm bảo rằng việc sử dụng của họ tuân thủ các giấy phép và/hoặc quyền đó.

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.