Linux
Professional
Institute

LPIC-1

Версія 5.0
Українська

102

Table of Contents

РОЗДІЛ 105: ОБОЛОНКИ ТА СЦЕНАРІЇ ОБОЛОНКИ . . . **1**

105.1 Налаштування та використання середовища оболонки . . . **2**

105.1 Урок 1 . . . 4

Вступ . . . 4

Типи оболонок: інтерактивні та неінтерактивні, оболонки з входом та оболонки без входу . . . 5

Вправи до посібника . . . 19

Дослідницькі вправи . . . 21

Підсумки . . . 23

Відповіді до вправ посібника . . . 25

Відповіді до дослідницьких вправ . . . 27

105.1 Урок 2 . . . 29

Вступ . . . 29

Змінні: присвоєння та звернення . . . 29

Локальні змінні або змінні оболонки . . . 33

Глобальні змінні або змінні середовища . . . 36

Вправи до посібника . . . 46

Дослідницькі вправи . . . 49

Підсумки . . . 51

Відповіді до вправ посібника . . . 53

Відповіді до дослідницьких вправ . . . 57

105.1 Урок 3 . . . 59

Вступ . . . 59

Створення псевдонимів . . . 59

Створення функцій . . . 64

Вправи до посібника . . . 74

Дослідницькі вправи . . . 77

Підсумки . . . 78

Відповіді до вправ посібника . . . 80

Відповіді до дослідницьких вправ . . . 85

105.2 Налаштування або написання простих сценаріїв . . . **86**

105.2 Урок 1 . . . 88

Вступ . . . 88

Структура та виконання сценарію . . . 89

Змінні . . . 91

Арифметичні вирази . . . 95

Умовне виконання . . . 95

Виведення сценарію . . . 97

Вправи до посібника . . . 99
Дослідницькі вправи . . . 100
Підсумки . . . 101
Відповіді до вправ посібника . . . 102
Відповіді до дослідницьких вправ . . . 103
105.2 Урок 2 . . . 104
Вступ . . . 104
Розширені тести . . . 104
Цикли . . . 110
Більш докладний приклад . . . 112
Вправи до посібника . . . 116
Дослідницькі вправи . . . 118
Підсумки . . . 119
Відповіді до вправ посібника . . . 120
Відповіді до дослідницьких вправ . . . 122
РОЗДІЛ 106: ІНТЕРФЕЙСИ КОРИСТУВАЧА ТА РОБОЧІ СТОЛИ . . . 123
106.1 Встановлення та налаштування X11 . . . 124
106.1 Урок 1 . . . 125
Вступ . . . 125
Архітектура системи X Window . . . 126
Конфігурація X-сервера . . . 129
Wayland . . . 134
Вправи до посібника . . . 136
Дослідницькі вправи . . . 137
Підсумки . . . 138
Відповіді до вправ посібника . . . 139
Відповіді до дослідницьких вправ . . . 140
106.2 Графічні робочі столи . . . 141
106.2 Урок 1 . . . 142
Вступ . . . 142
Система X Window . . . 143
Середовище робочого столу . . . 143
Популярні робочі середовища . . . 145
Взаємодія робочого столу . . . 147
Нелокальний доступ . . . 149
Вправи до посібника . . . 151
Дослідницькі вправи . . . 152
Підсумки . . . 153
Відповіді до вправ посібника . . . 154
Відповіді до дослідницьких вправ . . . 155

106.3 Доступність . . . **156**
106.3 Урок 1 . . . 157
Вступ . . . 157
Налаштування доступності . . . 157
Помічник для налаштування клавіатури та миші . . . 158
Порушення зору . . . 160
Вправи до посібника . . . 162
Дослідницькі вправи . . . 163
Підсумки . . . 164
Відповіді до вправ посібника . . . 165
Відповіді до дослідницьких вправ . . . 166
РОЗДІЛ 107: АДМІНІСТРАТИВНІ ЗАВДАННЯ . . . **167**
107.1 Керування обліковими записами користувачів і груп і відповідними системними файлами . . . **168**
107.1 Урок 1 . . . 170
Вступ . . . 170
Додавання облікових записів користувачів . . . 170
Змінення облікових записів користувачів . . . 172
Видалення облікових записів користувачів . . . 174
Додавання, зміна та видалення груп . . . 175
Шаблонний каталог . . . 175
Файл `/etc/login.defs` . . . 176
Команда `passwd` . . . 177
Команда `chage` . . . 178
Вправи до посібника . . . 180
Дослідницькі вправи . . . 181
Підсумки . . . 182
Відповіді до вправ посібника . . . 184
Відповіді до дослідницьких вправ . . . 186
107.1 Урок 2 . . . 189
Вступ . . . 189
/etc/passwd . . . 190
/etc/group . . . 191
/etc/shadow . . . 191
/etc/gshadow . . . 192
Фільтрація бази даних паролів і груп . . . 193
Вправи до посібника . . . 194
Дослідницькі вправи . . . 196
Підсумки . . . 197
Відповіді до вправ посібника . . . 198

Відповіді до дослідницьких вправ . . . 200
107.2 Автоматизація завдань системного адміністрування, планування завдань . . . 202
107.2 Урок 1 . . . 204
Вступ . . . 204
Планування завдання за допомогою Cron . . . 204
Користувацькі Crontabs . . . 205
Системні Crontabs . . . 206
Специфікації конкретного часу . . . 207
Змінні Crontab . . . 207
Створення завдань користувача Cron . . . 208
Створення системних завдань Cron . . . 209
Налаштування доступу до планування завдань . . . 210
Альтернатива Cron . . . 210
Вправи до посібника . . . 213
Дослідницькі вправи . . . 215
Підсумки . . . 216
Відповіді до вправ посібника . . . 217
Відповіді до дослідницьких вправ . . . 219
107.2 Урок 2 . . . 221
Вступ . . . 221
Планування роботи за допомогою `at` . . . 221
Список запланованих завдань за допомогою `atq` . . . 223
Видалення завдання за допомогою `atrm` . . . 223
Налаштуваня доступу до планування завдань . . . 224
Специфікації часу . . . 224
Альтернатива `at` . . . 224
Вправи до посібника . . . 226
Дослідницькі вправи . . . 227
Підсумки . . . 228
Відповіді до вправ посібника . . . 229
Відповіді до дослідницьких вправ . . . 230
107.3 Локалізація та інтернаціоналізація . . . 232
107.3 Урок 1 . . . 234
Вступ . . . 234
Часові пояси . . . 235
Літній час . . . 239
Мова та кодування символів . . . 240
Конвертація кодування . . . 243
Вправи до посібника . . . 245
Дослідницькі вправи . . . 246

Підсумки . . . 247
Відповіді до вправ посібника . . . 248
Відповіді до дослідницьких вправ . . . 249
РОЗДІЛ 108: ОСНОВНІ СИСТЕМНІ СЛУЖБИ . . . 250
108.1 Основні системні служби . . . 251
108.1 Урок 1 . . . 253
Вступ . . . 253
Місцевий та всесвітній час . . . 254
Дата . . . 254
Апаратний годинник . . . 256
timedatectl . . . 257
Налаштування часового поясу без timedatectl . . . 259
Встановлення дати й часу без timedatectl . . . 260
Вправи до посібника . . . 262
Дослідницькі вправи . . . 264
Підсумки . . . 265
Відповіді до вправ посібника . . . 267
Відповіді до дослідницьких вправ . . . 269
108.1 Урок 2 . . . 270
Вступ . . . 270
timedatectl . . . 272
NTP-демон . . . 273
Конфігурація NTP . . . 274
pool.ntp.org . . . 275
ntpdate . . . 275
ntpq . . . 275
chrony . . . 276
Вправи до посібника . . . 281
Дослідницькі вправи . . . 283
Підсумки . . . 284
Відповіді до вправ посібника . . . 285
Відповіді до дослідницьких вправ . . . 287
108.2 Системне журналювання . . . 288
108.2 Урок 1 . . . 290
Вступ . . . 290
Системне журналювання . . . 291
Вправи до посібника . . . 311
Дослідницькі вправи . . . 313
Підсумки . . . 314
Відповіді до вправ посібника . . . 315

Відповіді до дослідницьких вправ . . . 318
108.2 Урок 2 . . . 319
Вступ . . . 319
Основи `systemd` . . . 319
Системний журнал: `systemd-journald` . . . 320
Вправи до посібника . . . 339
Дослідницькі вправи . . . 342
Підсумки . . . 343
Відповіді до вправ посібника . . . 345
Відповіді до дослідницьких вправ . . . 348
108.3 Основи Mail Transfer Agent (MTA . . . 350
108.3 Урок 1 . . . 351
Вступ . . . 351
Локальний і віддалений MTA . . . 352
MTA для Linux . . . 353
Команда `mail` і поштові агенти користувача (MUA) . . . 359
Налаштування доставки . . . 360
Вправи до посібника . . . 362
Дослідницькі вправи . . . 363
Підсумки . . . 364
Відповіді до вправ посібника . . . 365
Відповіді до дослідницьких вправ . . . 366
108.4 Налаштування принтерів та друк . . . 367
108.4 Урок 1 . . . 368
Вступ . . . 368
Служба CUPS . . . 369
Встановлення принтера . . . 373
Керування принтерами . . . 376
Надсилання завдань друку . . . 377
Керування завданнями друку . . . 380
Видалення принтерів . . . 381
Вправи до посібника . . . 382
Дослідницькі вправи . . . 383
Підсумки . . . 384
Відповіді до вправ посібника . . . 386
Відповіді до дослідницьких вправ . . . 387
РОЗДІЛ 109: ОСНОВИ КОМП'ЮТЕРНИХ МЕРЕЖ . . . 389
109.1 Основи інтернет-протоколів . . . 390
109.1 Урок 1 . . . 391
Вступ . . . 391

IP (Інтернет протокол) . . . 391
Вправи до посібника . . . 401
Дослідницькі вправи . . . 402
Підсумки . . . 403
Відповіді до вправ посібника . . . 404
Відповіді до дослідницьких вправ . . . 405
109.1 Урок 2 . . . 406
Вступ . . . 406
Протокол керування передачею (TCP) . . . 408
Протокол дейтаграм користувача (UDP) . . . 408
Міжмережевий протокол керуючих повідомлень (ICMP) . . . 408
IPv6 . . . 409
Вправи до посібника . . . 412
Дослідницькі вправи . . . 413
Підсумки . . . 414
Відповіді до вправ посібника . . . 415
Відповіді до дослідницьких вправ . . . 416
109.2 Постійна конфігурація мережі . . . 417
109.2 Урок 1 . . . 418
Вступ . . . 418
Мережний інтерфейс . . . 418
Назви інтерфейсів . . . 420
Управління інтерфейсом . . . 421
Локальні та віддалені імена . . . 423
Вправи до посібника . . . 428
Дослідницькі вправи . . . 429
Підсумки . . . 430
Відповіді до вправ посібника . . . 431
Відповіді до дослідницьких вправ . . . 432
109.2 Урок 2 . . . 433
Вступ . . . 433
NetworkManager . . . 433
systemd-networkd . . . 438
Вправи до посібника . . . 441
Дослідницькі вправи . . . 442
Підсумки . . . 443
Відповіді до вправ посібника . . . 444
Відповіді до дослідницьких вправ . . . 445
109.3 Основи усунення несправностей мережі . . . 446
109.3 Урок 1 . . . 448

Вступ . . . 448
Про команду `ip` . . . 449
Огляд маски мережі та маршрутизації . . . 450
Налаштування інтерфейсу . . . 451
Таблиця маршрутизації . . . 453
Вправи до посібника . . . 457
Дослідницькі вправи . . . 458
Підсумки . . . 459
Відповіді до вправ посібника . . . 460
Відповіді до дослідницьких вправ . . . 462
109.3 Урок 2 . . . 464
Вступ . . . 464
Тестування з'єднань за допомогою `ping` . . . 464
Трасування маршрутів . . . 465
Пошук MTU за допомогою `tracepath` . . . 468
Створення довільних зв'язків . . . 469
Перегляд поточних підключень і слухачів . . . 470
Вправи до посібника . . . 472
Дослідницькі вправи . . . 473
Підсумки . . . 474
Відповіді до вправ посібника . . . 476
Відповіді до дослідницьких вправ . . . 478
109.4 Налаштування DNS на стороні клієнта . . . 480
109.4 Урок 1 . . . 481
Вступ . . . 481
Процес розпізнавання імен . . . 481
DNS класи . . . 482
Інструменти розпізнавання імен . . . 485
Вправи до посібника . . . 491
Дослідницькі вправи . . . 492
Підсумки . . . 493
Відповіді до вправ посібника . . . 494
Відповіді до дослідницьких вправ . . . 495
РОЗДІЛ 110: БЕЗПЕКА . . . 496
110.1 Виконання завдань з адміністрування безпеки . . . 497
110.1 Урок 1 . . . 499
Вступ . . . 499
Перевірка файлів із набором SUID і SGID . . . 499
Керування паролями та старіння . . . 502
Виявлення відкритих портів . . . 506

Обмеження на логіни користувачів, процеси та використання пам’яті . . . 512
Робота з зареєстрованими користувачами . . . 515
Базова конфігурація та використання `sudo` . . . 518
Вправи до посібника . . . 523
Дослідницькі вправи . . . 526
Підсумки . . . 527
Відповіді до вправ посібника . . . 529
Відповіді до дослідницьких вправ . . . 533
110.2 Налаштування безпеки хостау . . . 534
110.2 Урок 1 . . . 535
Вступ . . . 535
Покращення безпеки автентифікації за допомогою тіньових паролів . . . 535
Як використовувати Superdaemon для прослуховування вхідних мережних підключень . . . 537
Перевірка служб на непотрібні демони . . . 542
TCP обгортки як простий міжмережний екран . . . 544
Вправи до посібника . . . 545
Дослідницькі вправи . . . 546
Підсумки . . . 547
Відповіді до вправ посібника . . . 549
Відповіді до дослідницьких вправ . . . 550
110.3 Захист даних за допомогою шифрування . . . 551
110.3 Урок 1 . . . 553
Вступ . . . 553
Базова конфігурація та використання клієнта OpenSSH . . . 554
Роль ключів хоста сервера OpenSSH . . . 559
Тунелі SSH-портів . . . 561
Вправи до посібника . . . 566
Дослідницькі вправи . . . 568
Підсумки . . . 569
Відповіді до вправ посібника . . . 570
Відповіді до дослідницьких вправ . . . 573
110.3 Урок 2 . . . 574
Вступ . . . 574
Виконання базової конфігурації, використання та відкликання GnuPG . . . 575
Використовуйте GPG для шифрування, дешифрування, підпису та перевірки файлів . . . 581
Вправи до посібника . . . 586
Дослідницькі вправи . . . 588
Підсумки . . . 589

Відповіді до вправ посібника 590
Відповіді до дослідницьких вправ 592
Imprint **593**

Linux Professional Institute

Розділ 105: Оболонки та сценарії оболонки

105.1 Налаштування та використання середовища оболонки

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 105.1

Ваговий коефіцієнт

4

Ключові галузі знань

- Встановлення змінних середовища (наприклад, PATH) під час входу або під час створення нової оболонки.
- Написання функцій Bash для послідовностей команд, які часто використовуються.
- Підтримка шаблонних каталогів для нових облікових записів користувачів.
- Встановлення шляху пошуку команди з відповідним каталогом.

Частковий список файлів, термінів та утиліт, що використовуються

- `.`
- `source`
- `/etc/bash.bashrc`
- `/etc/profile`
- `env`
- `export`
- `set`
- `unset`
- `~/.bash_profile`
- `~/.bash_login`

- `~/.profile`
- `~/.bashrc`
- `~/.bash_logout`
- `function`
- `alias`

105.1 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	105 Оболонки та сценарії оболонки
Тема:	105.1
Урок:	1 з 3

Вступ

Оболонка, мабуть, є найпотужнішим інструментом у системі Linux і може бути визначена як інтерфейс між користувачем і ядром операційної системи. Вона інтерпретує команди, введені користувачем. Тому всі системні адміністратори повинні володіти навичками використання оболонки. Безсумнівно, оболонка Bourne Again Shell (*Bash*) є *де-факто* оболонкою для переважної більшості дистрибутивів Linux.

Після запуску перше, що робить Bash або будь-яка інша оболонка, це виконує серію сценаріїв запуску. Ці сценарії налаштовують середовище сеансу. Існують як системні, так і спеціальні сценарії. Ми можемо розмістити наші особисті вподобання або налаштування, які найкраще відповідають потребам наших користувачів, у цих сценаріях у формі змінних, псевдонімів і функцій.

Точна серія файлів запуску залежить від дуже важливого параметра: типу оболонки. Давайте розглянемо різні оболонки, які існують.

Типи оболонок: інтерактивні та неінтерактивні, оболонки з входом та оболонки без входу

Для початку давайте роз'яснимо поняття *інтерактивність* і *вхід* в контексті оболонок:

Інтерактивні/неінтерактивні оболонки

Цей вид оболонки відноситься до взаємодії, яка відбувається між користувачем і оболонкою: користувач забезпечує введення команди в терміналі за допомогою клавіатури; оболонка забезпечує виведення, друкуючи повідомлення на екрані.

Оболонки з входом/без входу

Цей вид оболонки відноситься до події, коли користувач отримує доступ до комп'ютерної системи, надаючи свої облікові дані, такі як ім'я користувача та пароль.

Як інтерактивні, так і неінтерактивні оболонки можуть потребувати входу в систему або не потребувати, і будь-яка можлива комбінація цих типів має своє конкретне використання.

Інтерактивні оболонки з входом виконуються, коли користувачі входять до системи, і вони використовуються для налаштування конфігурацій користувачів відповідно до їхніх потреб. Прикладом цього типу оболонки може бути група користувачів, що належать до одного відділу, яким потрібен певний набір змінних у їхніх сесіях.

Інтерактивними оболонками без входу назвемо будь-які інші оболонки, відкриті користувачем після входу в систему. Користувачі використовують ці оболонки під час сеансів для виконання завдань обслуговування та адміністративних завдань, таких як встановлення змінних, часу, копіювання файлів, написання сценаріїв тощо.

З іншого боку, *неінтерактивні оболонки* не потребують жодної взаємодії з людиною. Таким чином, ці оболонки не запитують у користувача вхідні дані, а їх вихідні дані - якщо такі є - у більшості випадків записуються до журналу.

Неінтерактивні оболонки з входом використовуються досить рідко та непрактичні. Їх майже не використовують, і ми будемо коментувати їх лише заради розуміння поведінки оболонки. Деякі дивні приклади включають примусовий запуск сценарію з оболонки з входом за допомогою `/bin/bash --login <some_script>` або передавання стандартного виводу (*stdout*) команди до стандартного входу (*stdin*) при ssh-з'єднанні:

```
<some_command> | ssh <some_user>@<some_server>
```

Що стосується *неінтерактивної оболонки без входу*, тут немає ні взаємодії, ні входу від імені користувача, тому ми маємо на увазі використання автоматизованих сценаріїв. Ці сценарії здебільшого використовуються для виконання завдань адміністрування та обслуговування, які повторюються, наприклад, включених в cronjobs. У таких випадках `bash` не читає файли запуску.

Відкриття терміналу

Коли ми перебуваємо в середовищі робочого столу, ми можемо або відкрити термінальну програму, або перейти до однієї з системних консолей. Таким чином, нова оболонка є або оболонкою `pts`, коли її відкривають із емулятора терміналу в графічному інтерфейсі користувача, або оболонкою `tty`, якщо її запускають із системної консолі. У першому випадку ми маємо справу не з терміналом, а з емулятором терміналу. Як частина графічних сеансів емулятори терміналів, такі як *gnome-terminal* або *konsole*, мають багато функцій та зручні для користувача порівняно з терміналами з текстовим інтерфейсом. Прикладом менш багатофункціональних емуляторів терміналу є *XTerm* і *sakura*.

Використовуючи комбінації `Ctrl` + `Alt` + `F1`-`F6`, ми можемо перейти до входу в консоль, яка відкриває інтерактивну текстову оболонку входу. `Ctrl` + `Alt` + `F7` поверне сеанс назад до робочого столу.

NOTE `tty` означає телетайп; `pts` означає псевдотермінал slave. Для отримання додаткової інформації скористайтеся командами `man tty` і `man pts`.

Запуск оболонок за допомогою `bash`

Після входу введіть `bash` у терміналі, щоб відкрити нову оболонку. Технічно ця оболонка є дочірнім процесом поточної оболонки.

Під час запуску дочірнього процесу bash ми можемо вказати різні опції, щоб визначити тип оболонки, яку ми хочемо запустити. Ось кілька важливих параметрів виклику bash:

`bash -l` або `bash --login`
: викличе оболонку входу.

`bash -i`
: викличе інтерактивну оболонку.

`bash --noprofile`
: з оболонками з входом ігноруватиме як загальносистемний файл запуску `/etc/profile`, так і файли запуску `~/.bash_profile`, `~/.bash_login` і `~/.profile` на

рівні користувача.

`bash --norc`

для оболонок з входом ігноруватиме як загальносистемний файл запуску `/etc/profile`, так і файли запуску `~/.bash_profile`, `~/.bash_login` і `~/.profile` на рівні користувача.

`bash --rcfile <file>`

для інтерактивних оболонок прийматиме `<file>` як файл запуску, ігноруючи системний `/etc/bash.bashrc` і `~/.bashrc` на рівні користувача.

Нижче ми обговоримо різні файли запуску.

Запуск оболонок за допомогою `su` і `sudo`

Завдяки використанню цих двох подібних програм ми можемо отримати певні типи оболонок:

`su`

Змініть ідентифікатор користувача або станьте суперкористувачем (`root`). За допомогою цієї команди ми можемо викликати оболонки з входом та без входу:

- `su - user2`, `su -l user2` або `su --login user2` запустить інтерактивну оболонку входу від імені `user2`.
- `su user2` запустить інтерактивну оболонку без входу від імені `user2`.
- `su - root` або `su -` запустить інтерактивну оболонку входу як `root`.
- `su root` або `su` запустить інтерактивну оболонку без входу як `root`.

`sudo`

Виконує команди від імені іншого користувача (включаючи суперкористувача). Оскільки ця команда в основному використовується для тимчасового отримання привілеїв root, користувач, який її використовує, має бути у файлі `sudoers`. Щоб додати користувачів до `sudoers`, нам потрібно стати `root`, а потім виконати команду:

```
root@debian:~# usermod -aG sudo user2
```

Подібно до `su`, `sudo` дозволяє нам викликати оболонки з входом та без входу:

- `sudo su - user2`, `sudo su -l user2` або `sudo su --login user2` запустить інтерактивну оболонку входу від імені `user2`.

- `sudo su user2` запустить інтерактивну оболонку без входу від імені `user2`.
- `sudo -u user2 -s` запустить інтерактивну оболонку без входу від імені `user2`.
- `sudo su - root` або `sudo su -` запустить інтерактивну оболонку входу як `root`.
- `sudo -i` запустить інтерактивну оболонку входу як `root`.
- `sudo -i <some_command>` запустить інтерактивну оболонку входу як `root`, запустить команду та повернеться до вихідного користувача.
- `sudo su root` або `sudo su` запустить інтерактивну оболонку без входу як `root`.
- `sudo -s` або `sudo -u root -s` запустить оболонку без входу як `root`.

Використовуючи `su` або `sudo`, важливо враховувати наш окремий сценарій запуску нової оболонки: чи потрібне нам середовище цільового користувача чи ні? Якщо так, ми б використали параметри, які викликають оболонки входу; якщо ні, то ті, які викликають оболонки без входу в систему.

Який тип оболонки запущено?

Щоб дізнатися, в якому типу оболонки ми працюємо, ми можемо ввести `echo $0` у термінал і отримати наступний результат:

Інтерактивний вхід

`-bash` або `-su`

Інтерактивний без входу

`bash` або `/bin/bash`

Неінтерактивний без входу (скрипти)

`<назва_скрипту>`

Скільки у нас оболонок?

Щоб побачити, скільки оболонок `bash` у нас запущено та працює в системі, ми можемо використати команду `ps aux | grep bash`:

```
user2@debian:~$ ps aux | grep bash
user2       5270  0.1  0.1  25532  5664 pts/0    Ss   23:03   0:00 bash
user2       5411  0.3  0.1  25608  5268 tty1     S+   23:03   0:00 -bash
user2       5452  0.0  0.0  16760   940 pts/0    S+   23:04   0:00 grep --color=auto bash
```

`user2` у `debian` увійшов до сеансу GUI (або X Window System) і відкрив *gnome-terminal*, потім він натиснув `Ctrl` + `Alt` + `F1`, щоб перейти до сеансу терміналу `tty`. Нарешті він повернувся до сеансу GUI, натиснувши `Ctrl` + `Alt` + `F7` і ввівши команду `ps aux | grep bash`. Таким чином, результат показує інтерактивну оболонку без входу через емулятор терміналу (`pts/0`) та інтерактивну оболонку з входом через відповідний текстовий термінал (`tty1`). Зауважте також, що останнє поле кожного рядка (команди) має значення `bash` для першого та `-bash` для останнього.

Звідки оболонки отримують свою конфігурацію: файли запуску

Що ж, тепер, коли ми знаємо типи оболонок, які ми можемо знайти в системі Linux, настав час побачити, які файли запуску запускаються відповідною оболонкою. Зауважте, що загальносистемні або глобальні сценарії розміщуються в каталозі `/etc/`, тоді як локальні або сценарії користувача знаходяться в його домашньому каталозі (`~`). Крім того, якщо шукати більше ніж один файл, після того, як один буде знайдено та запущено, інші ігноруватимуться. Досліджуйте та вивчайте ці файли самостійно за допомогою свого улюбленого текстового редактора або ввівши `less <startup_file>`.

NOTE Файли запуску можна розділити на спеціальні Bash-файли (обмежені лише конфігураціями та командами bash) і загальні (що стосуються більшості оболонок).

Інтерактивна оболонка входу

Глобальний рівень

`/etc/profile`

Це загальносистемний файл `.profile` для оболонки Bourne і сумісних з Bourne оболонок (включно з `bash`). За допомогою серії операторів if цей файл встановлює ряд змінних, таких як PATH і PS1 відповідно, а також вихідні коди (якщо вони існують) як у файлі /etc/bash.bashrc, так і в каталозі `/etc/profile.d`.

`/etc/profile.d/*`

Цей каталог може містити сценарії, які виконує `/etc/profile`.

Локальний рівень

`~/.bash_profile`

Цей специфічний файл Bash використовується для налаштування середовища користувача. Його також можна використовувати як джерело для `~/.bash_login` і `~/.profile`.

`~/.bash_login`

Цей специфічний для Bash файл буде виконано, лише якщо немає файлу `~/.bash_profile`. Назва передбачає, що його слід використовувати для запуску команд, необхідних під час входу.

`~/.profile`

Цей файл не є специфічним для Bash і використовується лише якщо не існує ні `~/.bash_profile`, ні `~/.bash_login` — що зазвичай і відбувається. Таким чином, основна мета `~/.profile` полягає в тому, щоб перевірити, чи запущена оболонка Bash, і, якщо так, знайти джерело `~/.bashrc`, якщо воно існує. Зазвичай він встановлює змінну PATH так, щоб вона містила приватний каталог користувача ~/bin, якщо він існує.

`~/.bash_logout`

Якщо цей специфічний файл Bash існує, він виконує деякі операції очищення під час виходу з оболонки. Це може бути зручно при віддалених сесіях.

Дослідження файлів конфігурації оболонки інтерактивного входу

Давайте покажемо деякі з цих файлів у дії, змінивши `/etc/profile` і `/home/user2/.profile`. Ми додамо до кожного рядок, який нагадуватиме про файл, який виконується:

```
root@debian:~# echo 'echo Hello from /etc/profile' >> /etc/profile
root@debian:~# echo 'echo Hello from ~/.profile' >> ~/.profile
```

NOTE Два оператори перенаправлення `>>` додають вихід команди до існуючого файлу, не перезаписуючи його. Однак якщо файл не існує, він буде створений.

Таким чином, за результатами відповідних команд `echo` ми дізнаємось, коли кожен із цих файлів буде прочитано та виконано. Щоб перевірити це, давайте подивимося, що відбувається, коли `user2` входить в систему через `ssh` з іншого комп'ютера:

```
user2@debian:~$ ssh user2@192.168.1.6
user2@192.168.1.6's password:
Linux debian 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Nov 27 19:57:19 2018 from 192.168.1.10
Hello from /etc/profile
Hello from /home/user2/.profile
```

Як показують останні два рядки, команди спрацювали. Також зверніть увагу на три речі:

- Глобальний файл був запущений спочатку.
- У домашньому каталозі `user2` не було файлів `.bash_profile` або `.bash_login`.
- Тильда (`~`) розширена до абсолютного шляху до файлу (`/home/user2/.profile`).

Інтерактивна оболонка без входу в систему

Глобальний рівень

`/etc/bash.bashrc`

Це загальносистемний файл `.bashrc` для інтерактивних оболонок `bash`. Під час виконання `bash` переконується, що він виконується в інтерактивному режимі, перевіряє розмір вікна після кожної команди (оновлюючи значення `LINES` і `COLUMNS`, якщо необхідно) і встановлює деякі змінні.

Локальний рівень

`~/.bashrc`

На додаток до виконання завдань, подібних до тих, що описані для `/etc/bash.bashrc` на рівні користувача (наприклад, перевірка розміру вікна або виконання в інтерактивному режимі), цей специфічний файл Bash зазвичай встановлює деякі змінні історії та джерела `~ /.bash_aliases`, якщо він існує. Окрім цього, цей файл зазвичай використовується для зберігання певних псевдонімів і функцій користувачів.

Крім того, варто зауважити, що `~/.bashrc` зчитується, якщо `bash` виявляє, що його `<stdin>` є мережевим підключенням (як це було у випадку з підключенням *Secure Shell* (SSH) у прикладі вище) .

Дослідження інтерактивних файлів конфігурації оболонки без входу в систему

Давайте тепер змінимо `/etc/bash.bashrc` і `/home/user2/.bashrc`:

```
root@debian:~# echo 'echo Hello from /etc/bash.bashrc' >> /etc/bash.bashrc
root@debian:~# echo 'echo Hello from ~/.bashrc' >> ~/.bashrc
```

І ось що відбувається, коли `user2` запускає нову оболонку:

```
user2@debian:~$ bash
Hello from /etc/bash.bashrc
Hello from /home/user2/.bashrc
```

Знову два файли були прочитані та виконані.

WARNING Пам'ятайте, що при запуску файлів локальні файли мають пріоритет над глобальними і запускаються першими.

Неінтерактивна оболонка входу

Неінтерактивна оболонка з параметрами `-l` або `--login` змушена вести себе як оболонка входу, тому файли запуску, які будуть запущені, будуть такими самими, як файли для інтерактивних оболонок входу.

Щоб перевірити це, давайте напишемо простий сценарій і зробимо його виконуваним. Ми не будемо включати жодних *shebangs*, оскільки ми будемо викликати *виконуваний файл_bash* (`/bin/bash` з опцією входу) з командного рядка.

1. Ми створюємо сценарій `test.sh`, що містить рядок `echo 'Hello from a script'`, щоб переконатися, що сценарій працює успішно:

   ```
   user2@debian:~$ echo "echo 'Hello from a script'" > test.sh
   ```

2. Ми робимо наш сценарій виконуваним:

   ```
   user2@debian:~$ chmod +x ./test.sh
   ```

3. Нарешті, ми викликаємо `bash` з опцією `-l`, щоб запустити сценарій:

   ```
   user2@debian:~$ bash -l ./test.sh
   Hello from /etc/profile
   Hello from /home/user2/.profile
   Hello from a script
   ```

Це працює! Перед запуском сценарію відбувся вхід і були виконані `/etc/profile` і `~/.profile`.

NOTE Ми дізнаємось про *shebangs* та всі інші аспекти сценаріїв оболонки в наступних уроках.

Тепер маємо стандартний вихід (*stdout*) команди `echo` у стандартному вході (*stdin*) з'єднання `ssh` за допомогою каналу (`|`):

```
user2@debian:~$ echo "Hello-from-a-noninteractive-login-shell" | ssh user2@192.168.1.6
Pseudo-terminal will not be allocated because stdin is not a terminal.
user2@192.168.1.6's password:
Linux debian 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Hello from /etc/profile
Hello from /home/user2/.profile
-bash: line 1: Hello-from-a-noninteractive-login-shell: command not found
```

Знову виконуються `/etc/profile` і `~/.profile`. Окрім цього, перший і останній рядки виведення досить інформативні щодо поведінки оболонки.

Неінтерактивна оболонка без входу

Сценарії не читають жоден із файлів, перелічених вище, але шукають змінну середовища `BASH_ENV`, розширюють її значення, якщо потрібно, і використовують як назву файлу запуску для читання та виконання команд. Ми дізнаємося більше про *змінні середовища* в наступному уроці.

Як згадувалося вище, зазвичай `/etc/profile` і `~/.profile` гарантують, що і `/etc/bash.bashrc`, і `~/.bashrc` виконуються після успішного входу. Виведення наступної команди показує це:

```
root@debian:~# su - user2
Hello from /etc/bash.bashrc
Hello from /etc/profile
Hello from /home/user2/.bashrc
```

```
Hello from /home/user2/.profile
```

Пам'ятаючи про рядки, які ми раніше додали до сценаріїв запуску – ,і викликаючи інтерактивну оболонку входу на рівні користувача за допомогою `su - user2`, чотири рядки виведення можна пояснити наступним чином:

1. `Hello from /etc/bash.bashrc` означає, що `/etc/profile` отримав дані файлу `/etc/bash.bashrc`.
2. `Hello from /etc/profile` означає, що `/etc/profile` повністю прочитано та виконано.
3. `Hello from /home/user2/.bashrc` означає, що `~/.profile` отримав дані файлу `~/.bashrc`.
4. `Hello from /home/user2/.profile` означає, що `~/.profile` повністю прочитано та виконано.

Зауважте, що за допомогою `su - <ім'я користувача>` (також `su -l <ім'я користувача>` і `su --login <ім'я користувача>`) ми гарантуємо виклик оболонки входу, тоді як `su <ім'я користувача>` лише викличе `/etc/bash.bashrc` і `~/.bashrc`.

Sourcing-файли

У попередніх розділах ми обговорювали, що деякі сценарії запуску включають або виконують інші сценарії. Цей механізм називається «sourcing» і пояснюється в цьому розділі.

Sourcing-файли з `.`

Крапка (`.`) зазвичай міститься у файлах запуску.

У файлі `.profile` нашого сервера Debian ми можемо знайти, наприклад, такий блок:

```
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
    . "$HOME/.bashrc"
    fi
```

Ми вже бачили, як виконання одного сценарію може призвести до виконання іншого. Таким чином, оператор if гарантує, що файл $HOME/.bashrc — якщо він існує (`-f`) — буде отримано (тобто прочитано та виконано) під час входу:

```
. "$HOME/.bashrc"
```

NOTE Як ми дізнаємося в наступному уроці, `$HOME` — це змінна середовища, яка розширюється до абсолютного шляху до домашнього каталогу користувача.

Крім того, ми можемо використовувати `.` щоразу, коли ми змінюємо файл запуску та хочемо зробити зміни дійсними без перезавантаження. Наприклад, ми можемо:

- додати псевдонім до `~/.bashrc`:

```
user2@debian:~$ echo "alias hi='echo We salute you.'" >> ~/.bashrc
```

WARNING Надсилаючи вихідні дані однієї команди у файл, не переплутайте додавання (`>>`) із перезаписом (`>`).

- виведіть останній рядок `~/.bashrc`, щоб перевірити, що все пройшло добре:

```
user2@debian:~$ tail -n 1 !$
tail -n 1 ~/.bashrc
alias hi='echo We salute you.'
```

NOTE `!$` розширюється до останнього аргументу з попередньої команди, у нашому випадку: `~/.bashrc`.

- вкажіть джерело файла вручну:

```
user2@debian:~$ . ~/.bashrc
```

- і викличте псевдонім, щоб довести, що він працює:

```
user2@debian:~$ hi
We salute you.
```

NOTE Зверніться до наступного уроку, щоб дізнатися про *псевдоніми* та *змінні*.

Sourcing-файли з використанням `source`

Команда `source` є синонімом `.`. Отже, щоб отримати source `~/.bashrc`, ми також можемо зробити це наступним чином:

```
user2@debian:~$ source ~/.bashrc
```

Походження файлів запуску оболонки: SKEL

`SKEL` — це змінна, значення якої є абсолютним шляхом до каталогу `skel`. Цей каталог служить шаблоном для структури файлової системи домашніх каталогів користувачів. Він містить файли, які будуть успадковані будь-якими новими створеними обліковими записами користувачів (включаючи, звичайно, файли конфігурації для оболонок). `SKEL` та інші пов'язані змінні зберігаються в `/etc/adduser.conf`, який є файлом конфігурації для `adduser`:

```
user2@debian:~$ grep SKEL /etc/adduser.conf
# The SKEL variable specifies the directory containing "skeletal" user
SKEL=/etc/skel
# If SKEL_IGNORE_REGEX is set, adduser will ignore files matching this
SKEL_IGNORE_REGEX="dpkg-(old|new|dist|save)"
```

`SKEL` має значення `/etc/skel`; таким чином, сценарії запуску, які налаштовують наші оболонки, містяться в ньому:

```
user2@debian:~$ ls -a /etc/skel/
.  ..  .bash_logout  .bashrc  .profile
```

WARNING Пам'ятайте, що файли, які починаються з `.`, приховані, тому ми повинні використовувати `ls -a`, щоб побачити їх під час перегляду вмісту каталогу.

Давайте тепер створимо каталог у `/etc/skel` для всіх нових користувачів для зберігання їх особистих сценаріїв:

1. Як `root` ми переходимо до `/etc/skel`:

```
root@debian:~# cd /etc/skel/
root@debian:/etc/skel#
```

2. Виводимо його вміст:

```
root@debian:/etc/skel# ls -a
```

```
.  ..  .bash_logout  .bashrc  .profile
```

3. Ми створюємо наш каталог і перевіряємо, чи все пройшло як очікувалося:

```
root@debian:/etc/skel# mkdir my_personal_scripts
root@debian:/etc/skel# ls -a
.  ..  .bash_logout  .bashrc  my_personal_scripts  .profile
```

4. Тепер ми видаляємо `user2` разом з його каталогом `home`:

```
root@debian:~# deluser --remove-home user2
Looking for files to backup/remove ...
Removing files ...
Removing user `user2' ...
Warning: group `user2' has no more members.
Done.
```

5. Ми знову додаємо `user2`, щоб отримати новий домашній каталог:

```
root@debian:~# adduser user2
Adding user `user2' ...
Adding new group `user2' (1001) ...
Adding new user `user2' (1001) with group `user2' ...
Creating home directory `/home/user2' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for user2
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
```

6. Нарешті, ми входимо як `user2` і виводимо всі файли в `/home/user2`, щоб побачити, чи все відбулося так, як очікувалося:

```
root@debian:~# su - user2
user2@debian:~$ pwd
/home/user2
user2@debian:~$ ls -a
.  ..  .bash_history  .bash_logout  .bashrc  my_personal_scripts  .profile
```

Саме так, як очікувалось.

Вправи до посібника

1. Дослідіть, як було запущено оболонки у стовпці “Запуск оболонки здійснено...” і додайте необхідну інформацію:

Запуск оболонки здійснено...	**Інтерактивно?**	**З входом?**	**Результат** `echo $0`
`sudo ssh user2@machine2`			
`Ctrl` + `Alt` + `F2`			
`su - user2`			
gnome-terminal			
Звичайний користувач використовує *konsole* для запуску екземпляру програми *sakura*			
Скрипт `test.sh` містить команду `echo $0`			

2. Напишіть команди `su` і `sudo`, щоб запустити вказану оболонку:

   Інтерактивна оболонка входу як `user2`

   `su`:

   `sudo`:

   Інтерактивна оболонка входу як `root`

   `su`:

   `sudo`:

   Інтерактивна оболонка без входу як `root`

   `su`:

   `sudo`:

Інтерактивна оболонка без входу як `user2`

`su`:

`sudo`:

3. Який файл запуску читається під час запуску оболонки зі стовпця “Тип оболонки”?

Тип оболонки	`/etc/profile`	`/etc/bash.bashrc`	`~/.profile`	`~/.bashrc`
Інтерактивна оболонка входу як `user2`				
Інтерактивна оболонка входу як `root`				
Інтерактивна оболонка без входу як `root`				
Інтерактивна оболонка без входу як `user2`				

Дослідницькі вправи

1. У Bash ми можемо написати просту функцію `Hello world!`, включивши такий код до порожнього файлу:

```
function hello() {
     echo "Hello world!"
}
```

   - Що ми повинні зробити далі, щоб зробити функцію доступною для оболонки?

   - Щойно вона стане доступною для поточної оболонки, як її викликати?

   - Для автоматизації, у який файл ви б помістили функцію та її виклик, щоб вона виконувалася, коли `user2` відкриває термінал із сеансу X Window? Який це тип оболонки?

   - У який файл ви б помістили функцію та її виклик, щоб вона запускалася, коли `root` запускає нову інтерактивну оболонку, незалежно від того, чи потребує вона вхід чи ні?

2. Подивіться на наступний базовий `bash` скрипт `Hello world!`:

```
#!/bin/bash

#hello_world: a simple bash script to discuss interaction in scripts.

echo "Hello world!"
```

   - Припустимо, ми робимо сценарій виконуваним і запускаємо його. Це буде інтерактивний сценарій? Чому?

   - Що робить сценарій інтерактивним?

3. Уявіть, що ви змінили значення деяких змінних у `~/.bashrc` і хочете, щоб ці зміни вступили в силу без перезавантаження. З вашого домашнього каталогу, як ви можете досягти цього двома різними способами?

4. Джон щойно розпочав сеанс X Window на сервері Linux. Він відкриває емулятор терміналу, щоб виконати деякі адміністративні завдання, але, на диво, сеанс зависає, і йому потрібно відкрити текстову оболонку.
   - Як він може відкрити цю `tty` -оболонку?
   - Які файли запуску будуть отримані?

5. Лінда є користувачем сервера Linux. Вона люб'язно просить адміністратора створити для неї файл `~/.bash_login`, щоб вона могла друкувати час і дату на екрані під час входу. Іншим користувачам подобається ця ідея, і вони наслідують її приклад. Адміністратору важко створити файл для всіх інших користувачів на сервері, тому він вирішує додати нову політику та створити `~/.bash_login` для всіх потенційних нових користувачів. Як адміністратор може виконати це завдання?

Підсумки

На цьому уроці ми дізналися:

- Оболонки встановлюють середовище користувача в системі Linux.
- *Bash* — це оболонка номер один для всіх дистрибутивів GNU/Linux.
- Перше завдання, яке виконує оболонка, це читання та виконання одного або кількох файлів запуску.
- Концепції *інтерактивності* і *входу* стосовно оболонок.
- Як запускати різні типи оболонок за допомогою `bash`, `su`, `sudo` і `Ctrl` + `Alt` + `F1`-`F6`.
- Як перевірити тип оболонки за допомогою `echo $0`.
- Локальні файли запуску `~/.bash_profile`, `~/.profile`, `~/.bash_login`, `~/.bash_logout` і `~/.bashrc`.
- Файли глобального запуску `/etc/profile`, `/etc/profile.d/*`, `/etc/bash.bashrc`.
- Локальні файли мають пріоритет над глобальними.
- Як перенаправити вихід команди за допомогою `>` (перезапис) і `>>` (додавання).
- Значення каталогу `skel`.
- Як отримати вихідні файли.

Команди, які використовуються в цьому уроці:

`bash`
: Створює нову оболонку.

`su`
: Створює нову оболонку.

`sudo`
: Створює нову оболонку.

`usermod`
: Змінює обліковий запис користувача.

`echo`
: Відображає рядок тексту.

`ps`

Виводить перелік поточних процесів.

`less`

Відображає довгі файли по сторінках.

`ssh`

Запускає підключення Open SSH (віддалено).

`chmod`

Змінює біти властивостей файлу, наприклад робить його виконуваним.

`grep`

Виводить рядки, що відповідають шаблону.

`ls`

Відображає вміст каталогу.

`cd`

Змінює каталог.

`mkdir`

Створює каталог.

`deluser`

Видаляє користувача.

`adduser`

Додає нового користувача.

`.`

Джерело файлу.

`source`

Джерело файлу.

`tail`

Виводить останню частину файлів.

Відповіді до вправ посібника

1. Дослідіть, як було запущено оболонки у стовпці “Запуск оболонки здійснено...” і додайте необхідну інформацію:

Запуск оболонки здійснено...	**Інтерактивно?**	**З входом?**	**Результат** `echo $0`
`sudo ssh user2@machine2`	Так	Так	`-bash`
`Ctrl` + `Alt` + `F2`	Так	Так	`-bash`
`su - user2`	Так	Так	`-bash`
gnome-terminal	Так	Ні	`bash`
Звичайний користувач використовує *konsole* для запуску екземпляру програми *sakura*	Так	Ні	`/bin/bash`
Скрипт `test.sh` містить команду `echo $0`	Ні	Ні	`./test.sh`

2. Напишіть команди `su` і `sudo`, щоб запустити вказану оболонку:

Інтерактивна оболонка входу як `user2`

`su`

`su - user2`, `su -l user2` or `su --login user2`

`sudo`

`sudo su - user2`, `sudo su -l user2` or `sudo su --login user2`

Інтерактивна оболонка входу як `root`

`su`

`su - root` or `su -`

`sudo`

`sudo su - root`, `sudo su -` or `sudo -i`

Інтерактивна оболонка без входу як `root`

`su`

`su root` or `su`

`sudo`

`sudo su root`, `sudo su`, `sudo -s` or `sudo -u root -s`

Інтерактивна оболонка без входу як `user2`

`su`

`su user2`

`sudo`

`sudo su user2` or `sudo -u user2 -s`

3. Який файл запуску читається під час запуску оболонки в розділі “Тип оболонки”?

Тип оболонки	`/etc/profile`	`/etc/bash.bashrc`	`~/.profile`	`~/.bashrc`
Інтерактивна оболонка входу як `user2`	Так	Так	Так	Так
Інтерактивна оболонка входу як `root`	Так	Так	Ні	Ні
Інтерактивна оболонка без входу як `root`	Ні	Так	Ні	Ні
Інтерактивна оболонка без входу як `user2`	Ні	Так	Ні	Так

Відповіді до дослідницьких вправ

1. У Bash ми можемо написати просту функцію `Hello world!`, включивши такий код до порожнього файлу:

```
function hello() {
     echo "Hello world!"
}
```

   - Що ми повинні зробити далі, щоб зробити функцію доступною для оболонки?

     Щоб зробити функцію доступною для поточної оболонки, ми повинні виконати source для файлу, який її містить.

   - Щойно вона стане доступною для поточної оболонки, як її викликати?

     Ми викличемо функцію, ввівши її назву в терміналі.

   - Для автоматизації, у який файл ви б помістили функцію та її виклик, щоб вона виконувалася, коли `user2` відкриває термінал із сеансу X Window? Який це тип оболонки?

     Найкращий файл для розміщення це `/home/user2/.bashrc`. Оболонка, що викликається, буде інтерактивною без входу.

   - У який файл ви б помістили функцію та її виклик, щоб вона запускалася, коли `root` запускає нову інтерактивну оболонку, незалежно від того, чи потребує вона входу чи ні?

     У `/etc/bash.bashrc`, оскільки цей файл виконується для всіх інтерактивних оболонок - незалежно від того, з входом вони чи ні.

2. Подивіться на наступний базовий `bash` скрипт `Hello world!`:

```
#!/bin/bash

#hello_world: a simple bash script to discuss interaction in scripts.

echo "Hello world!"
```

   - Припустимо, ми робимо сценарій виконуваним і запускаємо його. Це буде

інтерактивний сценарій? Чому?

Ні, оскільки немає взаємодії з людиною та команд, які вводить користувач.

- Що робить сценарій інтерактивним?

Той факт, що він вимагає введення користувача.

3. Уявіть, що ви змінили значення деяких змінних у `~/.bashrc` і хочете, щоб ці зміни вступили в дію без перезавантаження. З вашого домашнього каталогу, як ви можете досягти цього двома різними способами?

```
$ source .bashrc
```

або

```
$ . .bashrc
```

4. Джон щойно розпочав сеанс X Window на сервері Linux. Він відкриває емулятор терміналу, щоб виконати деякі адміністративні завдання, але, на диво, сеанс зависає, і йому потрібно відкрити текстову оболонку.

- Як він може відкрити цю `tty`-оболонку?

Він може зробити це, натиснувши `Ctrl` + `Alt` + `F1`-`F6`, щоб увійти в одну з шести оболонок tty.

- Які файли запуску будуть отримані?

`/etc/profile`
`/home/john/.profile`

5. Лінда є користувачем сервера Linux. Вона люб'язно просить адміністратора створити для неї файл `~/.bash_login`, щоб вона могла друкувати час і дату на екрані під час входу. Іншим користувачам подобається ця ідея, і вони наслідують її приклад. Адміністратору важко створити файл для всіх інших користувачів на сервері, тому він вирішує додати нову політику та створити `~/.bash_login` для всіх потенційних нових користувачів. Як адміністратор може виконати це завдання?

Він може досягти цього, помістивши `.bash_login` до каталогу `/etc/skel`.

105.1 Урок 2

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	105 Оболонки та сценарії оболонки
Тема:	105.1 Налаштування та використання середовища оболонки
Урок:	2 з 3

Вступ

Подумайте про змінну як про уявну коробку, в яку можна тимчасово помістити частину інформації. Так само, як і у сценаріях ініціалізації, Bash класифікує змінні як *shell/local* (ті, що живуть лише в межах оболонки, в якій вони були створені), або *environment/global* (ті, які успадковуються дочірніми оболонками та/або процесами). Фактично, на попередньому уроці ми розглянули оболонки та сценарії їх налаштування або ініціалізації. Зараз зручно зазначити, що сила цих файлів запуску полягає в тому, що вони дозволяють нам використовувати змінні, а також псевдоніми та функції, які допомагають нам створювати та налаштовувати середовище оболонки за нашим вибором.

Змінні: присвоєння та звернення

Змінна може бути визначена як ім’я, що містить значення.

У Bash надання значення імені називається *присвоєнням змінної*, і це спосіб, у який ми створюємо або встановлюємо змінні. З іншого боку, процес доступу до значення, що міститься в імені, називається *зверненням до змінної*.

Синтаксис для присвоєння значення змінній такий:

```
<ім'я_змінної>=<значення_змінної>
```

Наприклад:

```
$ distro=zorinos
```

Змінна `distro` дорівнює `zorinos`, тобто є частина пам'яті, яка містить значення `zorinos`, а `distro` є вказівником на цю частину пам'яті.

Однак зауважте, що під час присвоєння значення змінній не може бути жодного пробілу по обидва боки від знака рівності:

```
$ distro =zorinos
-bash: distro: command not found
$ distro= zorinos
-bash: zorinos: command not found
```

Через нашу помилку Bash прочитав `distro` і `zorinos` як команди.

Щоб звернутися до змінної (тобто перевірити її значення), ми використовуємо команду `echo`, перед назвою змінної ставимо знак `$`:

```
$ echo $distro
zorinos
```

Імена змінних

При виборі імен для змінних існують певні правила, які ми повинні враховувати.

Ім'я змінної може містити літери (`a-z`,`A-Z`), цифри (`0-9`) і підкреслення (`_`):

```
$ distro=zorinos
$ echo $distro
zorinos
$ DISTRO=zorinos
$ echo $DISTRO
zorinos
```

```
$ distro_1=zorinos
$ echo $distro_1
zorinos
$ _distro=zorinos
$ echo $_distro
zorinos
```

Ім’я не може починатися з числа, інакше Bash не зрозуміє це:

```
$ 1distro=zorinos
-bash: 1distro=zorinos: command not found
```

Воно не може містити пробілів (навіть лапки); за домовленістю замість них використовуються символи підкреслення:

```
$ "my distro"=zorinos
-bash: my: command not found
$ my_distro=zorinos
$ echo $my_distro
zorinos
```

Значення змінних

Стосовно значення змінних також важливо враховувати низку правил.

Змінні можуть містити будь-які буквено-цифрові символи (`a-z`,`A-Z`,`0-9`), а також більшість інших символів (`?`,`!`,`*`,`.`,`/` тощо):

```
$ distro=zorin12.4?
$ echo $distro
zorin12.4?
```

Значення змінних повинні бути взяті в лапки, якщо вони містять одинарні пробіли:

```
$ distro=zorin 12.4
-bash: 12.4: command not found
$ distro="zorin 12.4"
$ echo $distro
zorin 12.4
$ distro='zorin 12.4'
```

```
$ echo $distro
zorin 12.4
```

Значення змінних також повинні бути взяті в лапки, якщо вони містять символи, які використовуються для переспрямування (<,>) або символ вертикальної лінії (|). Єдине, що робить наступна команда, це створює порожній файл під назвою `zorin`:

```
$ distro=>zorin
$ echo $distro

$ ls zorin
zorin
```

Але це працює інакше, коли ми використовуємо лапки:

```
$ distro=">zorin"
$ echo $distro
>zorin
```

Однак одинарні та подвійні лапки не завжди взаємозамінні. Залежно від того, що ми робимо зі змінною (присвоєння чи звернення), використання того чи іншого виду лапок матиме наслідки та дасть різні результати. У контексті присвоєння значення змінним одинарні лапки розглядають символи значення змінної *буквально*, тоді як подвійні лапки дозволяють заміну змінної:

```
$ lizard=uromastyx
$ animal='My $lizard'
$ echo $animal
My $lizard
$ animal="My $lizard"
$ echo $animal
My uromastyx
```

З іншого боку, при зверненні до змінної, значення якої містить деякі початкові (або додаткові) пробіли, іноді поєднані із зірочками, ми обов'язково повинні використовувати подвійні лапки після команди `echo`, щоб уникнути *розбиття полів* і *розширення імені шляху*:

```
$ lizard="   genus   |   uromastyx"
```

```
$ echo $lizard
genus | uromastyx
$ echo "$lizard"
   genus   |   uromastyx
```

Якщо значення змінної містить знак оклику, що закривається, це має бути останній символ у рядку (інакше Bash подумає, що ми посилаємося на подію `history`):

```
$ distro=zorin.?/!os
-bash: !os: event not found
$ distro=zorin.?/!
$ echo $distro
zorin.?/!
```

Будь-які зворотні скісні риски повинні бути екрановані іншою зворотною скісною рискою. Крім випадку, якщо зворотний слеш є останнім символом у рядку, і ми його не екрануємо, Bash інтерпретує, що нам потрібен розрив рядка, і надасть нам новий рядок:

```
$ distro=zorinos\
>
$ distro=zorinos\\
$ echo $distro
zorinos\
```

У наступних двох розділах ми підсумуємо основні відмінності між *локальними* змінними і змінними *середовища*.

Локальні змінні або змінні оболонки

Локальні змінні або змінні оболонки існують лише в тій оболонці, в якій вони створені. За домовленістю локальні змінні записуються малими літерами.

Для проведення кількох тестів створимо локальну змінну. Як пояснювалося вище, ми вибираємо відповідну назву змінної та прирівнюємо її до відповідного значення. Наприклад:

```
$ reptile=tortoise
```

Давайте тепер скористаємося командою `echo` для звернення до нашої змінної та

перевіримо, чи все пройшло так, як очікувалося:

```
$ echo $reptile
tortoise
```

У певних сценаріях, наприклад, під час написання скриптів, незмінність може бути цікавою властивістю змінних. Якщо ми хочемо, щоб наші змінні були незмінними, ми можемо створити їх `лише для читання`:

```
$ readonly reptile=tortoise
```

Або надайте їм таку властивість після того, як вони були створені:

```
$ reptile=tortoise
$ readonly reptile
```

Тепер, якщо ми спробуємо змінити значення `reptile`, Bash відмовиться це зробити:

```
$ reptile=lizard
-bash: distro: readonly variable
```

NOTE Щоб отримати список усіх змінних лише для читання в нашому поточному сеансі, введіть `readonly` або `readonly -p` у терміналі.

Корисною командою під час роботи з локальними змінними є команда `set`.

`set` виводить усі поточні призначені змінні оболонки та функції. Оскільки це може бути багато рядків (спробуйте самі!), рекомендується використовувати її в поєднанні з командою, яка виводить інформацію частинами, наприклад, `less`:

```
$ set | less
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote:force_figno
re:histappend:interactive_comments:login_shell:progcomp:promptvars:sourcepath
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
BASH_CMDS=()
BASH_COMPLETION_COMPAT_DIR=/etc/bash_completion.d
```

```
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="4" [1]="4" [2]="12" [3]="1" [4]="release" [5]="x86_64-pc-linux-gnu")
BASH_VERSION='4.4.12(1)-release'
(...)
```

Чи існує наша змінна `reptile`?

```
$ set | grep reptile
reptile=tortoise
```

Так, ось вона!

Однак `reptile`, будучи локальною змінною, не буде успадкована жодними дочірніми процесами, створеними з поточної оболонки:

```
$ bash
$ set | grep reptile
$
```

І, звичайно, ми також не можемо вивести її значення:

```
$ echo $reptile
$
```

NOTE Ввівши команду bash у терміналі, ми відкриваємо нову (дочірню) оболонку.

Щоб скасувати будь-які змінні (локальні чи глобальні), ми використовуємо команду unset:

```
$ echo $reptile
tortoise
$ unset reptile
$ echo $reptile
$
```

NOTE Після `unset` потрібно зазначати лише ім'я змінної (без символу `$`).

Глобальні змінні або змінні середовища

Глобальні змінні або змінні середовища існують для поточної оболонки, а також для всіх наступних процесів, породжених з неї. За домовленістю змінні середовища записуються великими літерами:

```
$ echo $SHELL
/bin/bash
```

Ми можемо рекурсивно передати значення цих змінних іншим змінним, і значення останніх зрештою розшириться до значення перших:

```
$ my_shell=$SHELL
$ echo $my_shell
/bin/bash
$ your_shell=$my_shell
$ echo $your_shell
/bin/bash
$ our_shell=$your_shell
$ echo $our_shell
/bin/bash
```

Щоб локальна змінна оболонки стала змінною середовища, потрібно використати команду `export`:

```
$ export reptile
```

За допомогою `export reptile` ми перетворили нашу локальну змінну на змінну середовища, щоб дочірні оболонки могли розпізнавати її та використовувати:

```
$ bash
$ echo $reptile
tortoise
```

Подібним чином, `export` можна використовувати для присвоєння та експорту змінної одночасно:

```
$ export amphibian=frog
```

Тепер ми можемо відкрити новий екземпляр Bash і успішно звернутися до нової змінної:

```
$ bash
$ echo $amphibian
frog
```

NOTE За допомогою `export -n <ІМ'Я-ЗМІННОЇ>` змінна буде знову перетворена на локальну змінну оболонки.

Команда `export` також надасть нам перелік усіх існуючих змінних середовища, якщо ввести її без опцій (або з опцією `-p`):

```
$ export
declare -x HOME="/home/user2"
declare -x LANG="en_GB.UTF-8"
declare -x LOGNAME="user2"
(...)
declare -x PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
declare -x PWD="/home/user2"
declare -x SHELL="/bin/bash"
declare -x SHLVL="1"
declare -x SSH_CLIENT="192.168.1.10 49330 22"
declare -x SSH_CONNECTION="192.168.1.10 49330 192.168.1.7 22"
declare -x SSH_TTY="/dev/pts/0"
declare -x TERM="xterm-256color"
declare -x USER="user2"
declare -x XDG_RUNTIME_DIR="/run/user/1001"
declare -x XDG_SESSION_ID="8"
declare -x reptile="tortoise"
```

NOTE Команда `declare -x` еквівалентна `export`.

Ще дві команди, які можна використовувати для виведення списку всіх змінних середовища, це `env` і `printenv`:

```
$ env
SSH_CONNECTION=192.168.1.10 48678 192.168.1.7 22
LANG=en_GB.UTF-8
XDG_SESSION_ID=3
USER=user2
PWD=/home/user2
```

```
HOME=/home/user2
SSH_CLIENT=192.168.1.10 48678 22
SSH_TTY=/dev/pts/0
MAIL=/var/mail/user2
TERM=xterm-256color
SHELL=/bin/bash
SHLVL=1
LOGNAME=user2
XDG_RUNTIME_DIR=/run/user/1001
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
_=/usr/bin/env
```

Окрім того, що це синонім для `env`, ми іноді можемо використовувати `printenv` подібно до того, як використовуємо команду `echo` для перевірки значення змінної:

```
$ echo $PWD
/home/user2
$ printenv PWD
/home/user2
```

Однак зауважте, що з `printenv` перед назвою змінної не стоїть `$`.

NOTE `PWD` зберігає шлях поточного робочого каталогу. Про цю та інші загальні змінні середовища ми дізнаємося пізніше.

Запуск програми в модифікованому середовищі

`env` можна використовувати для зміни середовища оболонки під час виконання програми.

Щоб розпочати новий сеанс Bash із якомога порожнішим середовищем – очистивши більшість змінних (а також функцій і псевдонімів) – ми використаємо `env` з опцією `-i`:

```
$ env -i bash
```

Тепер більшість наших змінних середовища зникли:

```
$ echo $USER
$
```

І залишилося лише декілька:

```
$ env
LS_COLORS=
PWD=/home/user2
SHLVL=1
_=/usr/bin/printenv
```

Ми також можемо використовувати `env`, щоб встановити певну змінну для певної програми.

У нашому попередньому уроці під час обговорення *неінтерактивних оболонок без входу* ми бачили, як сценарії не читають жодних стандартних файлів запуску, а замість цього шукають значення змінної `BASH_ENV` і використовують це значення як файл запуску, якщо він існує.

Давайте продемонструємо цей процес:

1. Ми створюємо власний файл запуску під назвою `.startup_script` із таким вмістом:

   ```
   CROCODILIAN=caiman
   ```

2. Ми пишемо сценарій Bash під назвою `test_env.sh` з таким вмістом:

   ```
   #!/bin/bash

   echo $CROCODILIAN
   ```

3. Ми встановлюємо виконуваний біт для нашого сценарію `test_env.sh`:

   ```
   $ chmod +x test_env.sh
   ```

4. Нарешті, ми використовуємо `env`, щоб встановити `BASH_ENV` на `.startup_script` для `test_env.sh`:

   ```
   $ env BASH_ENV=/home/user2/.startup_script ./test_env.sh
   caiman
   ```

   Команда `env` є неявною, навіть якщо ми її позбудемося:

```
$ BASH_ENV=/home/user2/.startup_script ./test_env.sh
caiman
```

NOTE

Якщо ви не розумієте рядок `#!/bin/bash` або команду `chmod +x`, не панікуйте! У наступних уроках ми дізнаємося все необхідне про сценарії оболонки. Наразі пам'ятайте, що для виконання сценарію з його власного каталогу ми використовуємо `./some_script`.

Загальні змінні середовища

Настав час переглянути деякі з найбільш релевантних змінних середовища, встановлених у файлах конфігурації Bash.

`DISPLAY`

Щодо X-сервера, значення цієї змінної зазвичай складається з трьох елементів:

- Ім'я хоста (його відсутність означає `localhost`), на якому працює X-сервер.
- Двокрапка як роздільник.
- Число (зазвичай це `0` і відноситься до дисплея комп'ютера).

```
$ printenv DISPLAY
:0
```

Порожнє значення для цієї змінної означає сервер без системи X Window. Додаткове число, як у `my.xserver:0:1`, посилатиметься на номер екрана, якщо існує більше одного.

`HISTCONTROL`

Ця змінна визначає, які команди зберігаються у `HISTFILE` (див. нижче). Існує три можливі значення:

`ignorespace`

Команди, що починаються з пробілу, не будуть збережені.

`ignoredups`

Команда, яка збігається з попередньою, не буде збережена.

`ignoreboth`

Команди, які належать до будь-якої з двох попередніх категорій, не будуть збережені.

```
$ echo $HISTCONTROL
ignoreboth
```

HISTSIZE

Встановлює кількість команд, які будуть зберігатися в пам'яті під час сеансу оболонки.

```
$ echo $HISTSIZE
1000
```

HISTFILESIZE

Встановлює кількість команд, які будуть збережені у `HISTFILE` як на початку, так і наприкінці сеансу:

```
$ echo $HISTFILESIZE
2000
```

HISTFILE

Ім'я файлу, у якому зберігаються всі команди під час їх введення. За замовчуванням цей файл знаходиться в `~/.bash_history`:

```
$ echo $HISTFILE
/home/user2/.bash_history
```

NOTE Щоб переглянути вміст `HISTFILE`, ми просто вводимо `history`. Крім того, ми можемо вказати кількість команд, які ми хочемо бачити, передавши аргумент (кількість останніх команд) у `history`, наприклад, `history 3`.

HOME

Ця змінна зберігає абсолютний шлях до домашнього каталогу поточного користувача та встановлюється під час входу користувача.

Цей фрагмент коду, починаючи з `~/.profile`, не потребує пояснень (він є джерелом `"$HOME/.bashrc"`, якщо він існує):

```
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
    . "$HOME/.bashrc"
```

```
    fi
```

NOTE Якщо ви не зовсім розумієте оператор if, не хвилюйтеся: просто зверніться до уроків про сценарії оболонки.

Пам'ятайте, що `~` еквівалентно `$HOME`:

```
$ echo ~; echo $HOME
/home/carol
/home/carol
```

NOTE Команди можна об'єднати крапкою з комою (`;`).

Ми також можемо підтвердити це за допомогою оператора if:

```
$ if [ ~ == "$HOME" ]; then echo "true"; else echo "false"; fi
true
```

NOTE Пам'ятайте: знак рівності `=` використовується для . `==` використовується для перевірки рівності.

HOSTNAME

Ця змінна зберігає TCP/IP ім'я хост-комп'ютера:

```
$ echo $HOSTNAME
debian
```

HOSTTYPE

Тут зберігається інформація про архітектуру процесора хост-комп'ютера:

```
$ echo $HOSTTYPE
x86_64
```

LANG

Ця змінна зберігає локаль (регіональні налаштування) системи:

```
$ echo $LANG
en_UK.UTF-8
```

`LD_LIBRARY_PATH`

Ця змінна складається з розділеного двокрапкою набору каталогів, де загальні бібліотеки спільно використовуються програмами:

```
$ echo $LD_LIBRARY_PATH
/usr/local/lib
```

`MAIL`

Ця змінна зберігає файл, у якому Bash шукає електронну пошту:

```
$ echo $MAIL
/var/mail/carol
```

Іншим загальним значенням для цієї змінної є `/var/spool/mail/$USER`.

`MAILCHECK`

Ця змінна зберігає числове значення, яке вказує частоту в секундах, з якою Bash перевіряє наявність нової пошти:

```
$ echo $MAILCHECK
60
```

`PATH`

Ця змінна середовища зберігає список каталогів, у яких Bash шукає виконувані файли, коли йому доручається запустити будь-яку програму. Для комп'ютера в нашому прикладі ця змінна встановлюється через загальносистемний файл `/etc/profile`:

```
if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
fi
export PATH
```

За допомогою оператора if перевіряється ідентичність користувача, і залежно від результату перевірки (root чи інший) ми отримаємо той чи інший PATH. Нарешті вибраний `PATH` поширюється за допомогою `export`.

Зверніть увагу на дві речі щодо значення PATH:

- Імена каталогів записуються з використанням абсолютних шляхів.
- Двокрапка використовується як роздільник.

Якщо ми хочемо включити папку `/usr/local/sbin` в `PATH` для звичайних користувачів, ми змінимо рядок, щоб він виглядав наступним чином:

```
(...)
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/usr/local/sbin"
fi
export PATH
```

Тепер ми можемо побачити, як змінюється значення змінної, коли ми входимо як звичайний користувач:

```
# su - carol
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/usr/local/sbin
```

NOTE

Ми також могли додати `/usr/local/sbin` до `PATH` користувача в командному рядку, ввівши `PATH=/usr/local/sbin:$PATH` або `PATH=$PATH:/usr/local /sbin` — перший запис робить `/usr/local/sbin` першим каталогом для пошуку виконуваних файлів; останній робить його останнім.

PS1

Ця змінна зберігає значення підказки Bash. У наступному фрагменті коду (також із `/etc/profile`) оператор `if` перевіряє особу користувача та надає йому дуже скорочену підказку відповідно (`#` для `root` або `$` для постійних користувачів):

```
if [ "`id -u`" -eq 0 ]; then
  PS1='# '
else
  PS1='$ '
fi
```

NOTE

`id` для `root` дорівнює 0. Перемкніться на `root` і перевірте це самостійно за

> допомогою `id -u`.

Інші змінні запиту включають:

`PS2`
: зазвичай має значення `>` і використовується як підказка продовження для довгих багаторядкових команд.

`PS3`
: Використовується як підказка для команди `select`.

`PS4`
: зазвичай встановлюється на `+` та використовується для налагодження.

`SHELL`

Ця змінна зберігає абсолютний шлях поточної оболонки:

```
$ echo $SHELL
/bin/bash
```

`USER`

Тут зберігається ім’я поточного користувача:

```
$ echo $USER
carol
```

Вправи до посібника

1. Зверніть увагу на призначення змінних у стовпці “Команда(и)” та вкажіть, чи є отримана змінна “Локальною” чи “Глобальною”:

Команда(и)	Локальна	Глобальна
`debian=mother`		
`ubuntu=deb-based`		
`mint=ubuntu-based; export mint`		
`export suse=rpm-based`		
`zorin=ubuntu-based`		

2. Розгляньте поля “Команда” і “Виведення” та поясніть значення:

Команда	Виведення	Значення
`echo $HISTCONTROL`	`ignoreboth`	
`echo ~`	`/home/carol`	
`echo $DISPLAY`	`reptilium:0:2`	
`echo $MAILCHECK`	`60`	
`echo $HISTFILE`	`/home/carol/.bash_history`	

3. Змінні встановлюються неправильно в стовпці “Неправильна команда”. Надайте відсутню інформацію в розділах “Правильна команда” і “Посилання на змінну”, щоб ми отримали “Очікуваний результат”:

Неправильна команда	Правильна команда	Посилання на змінну	Очікуваний результат
`lizard =chameleon`			`chameleon`
`cool lizard=chameleon`			`chameleon`
`lizard=cha|me|leon`			`cha|me|leon`

Неправильна команда	**Правильна команда**	**Посилання на змінну**	**Очікуваний результат**
`lizard=/** chameleon **/`			`/** chamelon **/`
`win_path=C:\path\to\dir\`			`C:\path\to\dir\`

4. Розгляньте мету та напишіть відповідну команду:

Мета	**Команда**
Встановіть мову поточної оболонки на іспанську UTF-8 (`es_ES.UTF-8`).	
Виведіть назву поточного робочого каталогу.	
Зверніться до змінної середовища, яка зберігає інформацію про з'єднання `ssh`.	
Встановіть `PATH`, щоб включити `/home/carol/scripts` як останній каталог для пошуку виконуваних файлів.	
Встановіть для `my_path` значення `PATH`.	
Встановіть значення `my_path` на значення `PATH`.	

5. Створіть локальну змінну з назвою `mammal` і призначте їй значення `gnu`:

6. Використовуючи підстановку змінної, створіть ще одну локальну змінну з назвою `var_sub` з відповідним значенням, щоб при посиланні через `echo $var_sub` ми отримували: `The value of mammal is gnu`:

7. Перетворіть `mammal` на змінну середовища:

8. Знайдіть її за допомогою `set` і `grep`:

9. Знайдіть її за допомогою `env` і `grep`:

10. Створіть двома послідовними командами змінну середовища з іменем `BIRD` зі значенням `penguin`:

11. Створіть за допомогою однієї команди змінну середовища з назвою `NEW_BIRD` зі значенням `yellow-eyed penguin`:

12. Якщо ви `user2`, створіть папку з назвою `bin` у вашому домашньому каталозі:

13. Введіть команду, щоб додати папку `~/bin` до вашого `PATH`, щоб це був перший каталог, у якому `bash` шукає двійкові файли:

14. Щоб гарантувати, що значення `PATH` залишиться незмінним під час перезавантаження, який фрагмент коду у формі оператора `if` ви б додали до `~/.profile`?

Дослідницькі вправи

1. `let`: більше ніж отримання арифметичного виразу:
   - Виконайте пошук на сторінці довідки або в Інтернеті для команди `let` і її результатів під час встановлення змінних і створіть нову локальну змінну з назвою `my_val` зі значенням `10` як результат додавання 5 і 5:
   - Тепер створіть ще одну змінну під назвою `your_val` зі значенням `5` як результат ділення значення `my_val` на 2:
2. Результат виконання команди в змінній? Звичайно, це можливо; це називається *заміна команд*. Дослідіть цю можливість та дослідіть функцію під назвою `music_info`:

```
music_info(){
latest_music=`ls -l1t ~/Music | head -n 6`
echo -e "Your latest 5 music files:\n$latest_music"
}
```

Результат виконання команди `ls -l1t ~/Music | head -n 6` стає значенням змінної `latest_music`. Потім змінна `latest_music` виводиться командою `echo` (виводить загальну кількість байтів, зайнятих папкою `Music`, і останні п'ять музичних файлів, що зберігаються в папці `Music`, по одному на рядок).

Що з наведеного нижче є дійсним синонімом для

```
latest_music=`ls -l1t ~/Music | head -n 6`
```

Опція A:

```
latest_music=$(ls -l1t ~/Music| head -n 6)
```

Опція B:

```
latest_music="(ls -l1t ~/Music| head -n 6)"
```

Опція C:

```
latest_music=((ls -l1t ~/Music| head -n 6))
```

Підсумки

В цьому уроці ми вивчили:

- Що змінні є дуже важливою частиною середовища оболонки, оскільки вони використовуються як самою оболонкою, так і іншими програмами.
- Як призначати значення змінним та звертатися до змінних.
- Відмінності між *локальними* і *глобальними* (або *змінними середовища*) змінними.
- Як зробити змінні *readonly*.
- Як перетворити локальну змінну на змінну середовища за допомогою команди `export`.
- Як вивести всі змінні середовища.
- Як запустити програму в модифікованому середовищі.
- Як зробити змінні постійними за допомогою сценаріїв запуску.
- Деякі загальні змінні середовища: `DISPLAY`, `HISTCONTROL`, `HISTSIZE`, `HISTFILESIZE`, `HISTFILE`, `HOME`, `HOSTNAME`, `HOSTTYPE`, `LANG`, `LD_LIBRARY_PATH`, `MAIL`, `MAILCHECK`, `PATH`, `PS1` (та інші змінні підказки), `SHELL` і `USER`.
- Значення тильди (`~`).
- Основи оператору if.

Команди, які використані в цьому уроці:

`echo`

Звернення до змінної.

`ls`

Список вмісту каталогу.

`readonly`

Робить змінні незмінними. Виводить усі змінні лише для читання в поточному сеансі.

`set`

Виводить усі змінні і функції у поточному сеансі.

`grep`

Виводить рядки, що відповідають шаблону.

`bash`

Запускає нову оболонку

`unset`

Видаляє змінні.

`export`

Перетворює локальну змінну на змінну середовища. Перелік змінних середовища.

`env`

Перелік змінних середовища. Запуск програми в модифікованому середовищі.

`printenv`

Перелік змінних середовища. Звернення до змінної.

`chmod`

Змінює біти властивостей файлу, наприклад робить його виконуваним.

`history`

Виводить перелік попередніх команд.

`su`

Змінює ідентифікатор користувача або встановлює суперкористувача.

`id`

Виводить ідентифікатор користувача.

Відповіді до вправ посібника

1. Зверніть увагу на призначення змінних у стовпці “Команда(и)” та вкажіть, чи є отримана змінна “Локальною” чи “Глобальною”:

Команда(и)	**Локальна**	**Глобальна**
`debian=mother`	Так	Ні
`ubuntu=deb-based`	Так	Ні
`mint=ubuntu-based; export mint`	Ні	Так
`export suse=rpm-based`	Ні	Так
`zorin=ubuntu-based`	Так	Ні

2. Розгляньте поля “Команда” і “Виведення” та поясніть значення:

Команда	**Виведення**	**Значення**
`echo $HISTCONTROL`	`ignoreboth`	Команди, що дублюються, та команди, що починаються з пробілу, не будуть збережені в `history`.
`echo ~`	`/home/carol`	`HOME` для `carol` є `/home/carol`.
`echo $DISPLAY`	`reptilium:0:2`	На комп’ютері `reptilium` працює X-сервер, і ми використовуємо другий екран дисплея.
`echo $MAILCHECK`	`60`	Пошта буде перевірятися щохвилини.
`echo $HISTFILE`	`/home/carol/.bash_history`	`history` буде збережена в `/home/carol/.bash_history`.

3. Змінні встановлюються неправильно в стовпці “Неправильна команда”. Надайте відсутню інформацію в розділах “Правильна команда” і “Посилання на змінну”, щоб ми отримали “Очікуваний результат”:

Неправильна команда	**Правильна команда**	**Посилання на змінну**	**Очікуваний результат**
`lizard =chameleon`	`lizard=chameleon`	`echo $lizard`	`chameleon`
`cool lizard=chameleon`	`cool_lizard=chameleon` (наприклад)	`echo $cool_lizard`	`chameleon`
`lizard=cha|me|leon`	`lizard="cha|me|leon"` або `lizard='cha|me|leon'`	`echo $lizard`	`cha|me|leon`
`lizard=/** chameleon **/`	`lizard="/** chameleon **/"` або `lizard='/** chameleon **/'`	`echo "$lizard"`	`/** chamelon **/`
`win_path=C:\path\to\dir\`	`win_path=C:\\path\\to\\dir\\`	`echo $win_path`	`C:\path\to\dir\`

4. Розгляньте мету та напишіть відповідну команду:

Мета	**Команда**
Встановіть мову поточної оболонки на іспанську UTF-8 (`es_ES.UTF-8`).	`LANG=es_ES.UTF-8`
Виведіть назву поточного робочого каталогу.	`echo $PWD` або `pwd`
Зверніться до змінної середовища, яка зберігає інформацію про з'єднання `ssh`.	`echo $SSH_CONNECTION`
Встановіть `PATH`, щоб включити `/home/carol/scripts` як останній каталог для пошуку виконуваних файлів.	`PATH=$PATH:/home/carol/scripts`
Встановіть для `my_path` значення `PATH`.	`my_path=PATH`
Встановіть значення `my_path` на значення `PATH`.	`my_path=$PATH`

5. Створіть локальну змінну з назвою `mammal` і призначте їй значення `gnu`:

```
mammal=gnu
```

6. Використовуючи підстановку змінної, створіть ще одну локальну змінну з назвою `var_sub` з відповідним значенням, щоб при посиланні через `echo $var_sub` ми отримували: `The value of mammal is gnu`:

```
var_sub="The value of mammal is $mammal"
```

7. Перетворіть `mammal` на змінну середовища:

```
export mammal
```

8. Знайдіть її за допомогою `set` і `grep`:

```
set | grep mammal
```

9. Знайдіть її за допомогою `env` і `grep`:

```
env | grep mammal
```

10. Створіть двома послідовними командами змінну середовища з іменем `BIRD` зі значенням `penguin`:

```
BIRD=penguin; export BIRD
```

11. Створіть за допомогою однієї команди змінну середовища з назвою `NEW_BIRD` зі значенням `yellow-eyed penguin`:

```
export NEW_BIRD="yellow-eyed penguin"
```

або

```
export NEW_BIRD='yellow-eyed penguin'
```

12. Якщо ви `user2`, створіть папку з назвою `bin` у вашому домашньому каталозі:

```
mkdir ~/bin
```

або

```
mkdir /home/user2/bin
```

або

```
mkdir $HOME/bin
```

13. Введіть команду, щоб додати папку `~/bin` до вашого `PATH`, щоб це був перший каталог, у якому `bash` шукає двійкові файли:

```
PATH="$HOME/bin:$PATH"
```

`PATH=~/bin:$PATH` або `PATH=/home/user2/bin:$PATH` однаково правильні.

14. Щоб гарантувати, що значення `PATH` залишиться незмінним під час перезавантаження, який фрагмент коду у формі оператора `if` ви б додали до `~/.profile`?

```
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

Відповіді до дослідницьких вправ

1. `let`: більше ніж отримання арифметичного виразу:
   - Виконайте пошук на сторінці довідки або в Інтернеті для команди `let` і її результатів під час встановлення змінних і створіть нову локальну змінну з назвою `my_val` зі значенням `10` як результат додавання 5 і 5:

     ```
     let "my_val = 5 + 5"
     ```

     або

     ```
     let 'my_val = 5 + 5'
     ```

   - Тепер створіть ще одну змінну під назвою `your_val` зі значенням `5` як результат ділення значення `my_val` на 2:

     ```
     let "your_val = $my_val / 2"
     ```

     або

     ```
     let 'your_val = $my_val / 2'
     ```

2. Результат виконання команди в змінній? Звичайно, це можливо; це називається *заміна команд*. Дослідіть цю можливість та дослідіть функцію під назвою `music_info`:

   ```
   music_info(){
   latest_music=`ls -l1t ~/Music | head -n 6`
   echo -e "Your latest 5 music files:\n$latest_music"
   }
   ```

   Результат виконання команди `ls -l1t ~/Music | head -n 6` стає значенням змінної `latest_music`. Потім змінна `latest_music` виводиться командою `echo` (виводить загальну кількість байтів, зайнятих папкою `Music`, і останні п’ять музичних файлів, що зберігаються в папці `Music`, по одному на рядок).

   Що з наведеного нижче є дійсним синонімом для

```
latest_music=`ls -l1t ~/Music | head -n 6`
```

Це опція A:

```
latest_music=$(ls -l1t ~/Music| head -n 6)
```

105.1 Урок 3

Сетрифікат:	LPIC-1
Версія:	5.0
Розділ:	105 Оболонки та сценарії оболонки
Тема:	105.1 Налаштування та використання середовища оболонки
Урок:	3 з 3

Вступ

Після ознайомлення з оболонками, сценаріями запуску та змінними в попередніх уроках ми завершимо всю тему налаштування оболонки, поглянувши на два дуже цікаві елементи оболонки: *псевдоніми* і *функції*. Насправді вся група змінних, псевдонімів і функцій, і їхній вплив однн на одного, це те, що складає середовище оболонки.

Основна перевага цих двох гнучких засобів оболонки, економія часу, пов'язана з концепцією інкапсуляції: вони пропонують можливість об'єднати, в одній команді, серію повторюваних або рекурентних команд.

Створення псевдонимів

Псевдонім – це ім'я, яке замінює іншу команду(и). Він може працювати як звичайна команда, але натомість виконує іншу команду відповідно до визначення псевдоніма.

Синтаксис для оголошення псевдонімів досить простий. Псевдоніми оголошуються записом ключового слова `alias`, за яким слідує призначення псевдоніма. У свою чергу,

призначення псевдоніма складається з *псевдоніма, знаку рівності* та однієї або кількох *команд*:

```
alias alias_name=command(s)
```

Наприклад:

```
$ alias oldshell=sh
```

Цей незгарбний псевдонім запускає екземпляр оригінальної оболонки `sh`, коли користувач вводить `oldshell` у терміналі:

```
$ oldshell
$
```

Сила псевдонімів полягає в тому, що вони дозволяють нам писати короткі версії довгих команд:

```
$ alias ls='ls --color=auto'
```

NOTE Щоб отримати інформацію про команду ls та його кольори, введіть `man dir_colors` у терміналі.

Подібним чином ми можемо створювати псевдоніми для серії об'єднаних команд – крапка з комою (`;`) використовується як роздільник. Ми можемо, наприклад, мати псевдонім, який дає нам інформацію про розташування виконуваного файлу `git` та його версію:

```
$ alias git_info='which git;git --version'
```

Щоб викликати псевдонім, ми вводимо його назву в терміналі:

```
$ git_info
/usr/bin/git
git version 2.7.4
```

Команда `alias` виведе список усіх доступних псевдонімів у системі:

```
$ alias
alias git-info='which git;git --version'
alias ls='ls --color=auto'
alias oldshell='sh'
```

Команда `unalias` видаляє псевдоніми. Ми можемо, наприклад, виконати `unalias git-info` і подивитися, як він зникає зі списку:

```
$ unalias git-info
$ alias
alias ls='ls --color=auto'
alias oldshell='sh'
```

Як ми бачили з `alias hi='echo We salute you.'` у попередньому уроці, ми повинні брати команди в лапки (одинарні або подвійні), якщо вони містять пробіли через наявність аргументів або параметрів. :

```
$ alias greet='echo Hello world!'
$ greet
Hello world!
```

Команди з пробілами включають також команди з параметрами:

```
$ alias ll='ls -al'
```

Тепер `ll` покаже список усіх файлів, включно з прихованими (`a`), у довгому форматі (`l`).

Ми можемо посилатися на змінні в псевдонімах:

```
$ reptile=uromastyx
$ alias greet='echo Hello $reptile!'
$ greet
Hello uromastyx!
```

Змінній також можна призначити значення в псевдонімі:

```
$ alias greet='reptile=tortoise; echo Hello $reptile!'
$ greet
```

```
Hello tortoise!
```

Ми можемо екранувати псевдонім за допомогою `\`:

```
$ alias where?='echo $PWD'
$ where?
/home/user2
$ \where?
-bash: where?: command not found
```

Екранування псевдоніма корисно, якщо псевдонім має таке ж ім'я, як і звичайна команда. У цьому випадку псевдонім має перевагу над оригінальною командою, яка, однак, все одно доступна, якщо екранувати псевдонім.

Так само ми можемо розмістити псевдонім всередині іншого псевдоніма:

```
$ where?
/home/user2
$ alias my_home=where?
$ my_home
/home/user2
```

Крім того, ми також можемо розмістити функцію в псевдонімі, як буде показано нижче.

Розширення та оцінювання лапок у псевдонімах

При використанні лапок зі змінними середовища одинарні лапки роблять розширення динамічним:

```
$ alias where?='echo $PWD'
$ where?
/home/user2
$ cd Music
$ where?
/home/user2/Music
```

Однак у подвійних лапках розширення виконується статично:

```
$ alias where?="echo $PWD"
$ where?
```

```
/home/user2
$ cd Music
$ where?
/home/user2
```

Постійність псевдонімів: сценарії запуску

Так само, як і зі змінними, щоб наші псевдоніми стали постійними, ми повинні помістити їх у сценарії ініціалізації, які отримуються під час запуску. Як ми вже знаємо, зручним файлом для користувачів, щоб додати свої особисті псевдоніми є `~/.bashrc`. Ймовірно, ви вже знайдете деякі псевдоніми (більшість із них закоментовано та готові до використання, якщо видалити `#` на початку):

```
$ grep alias .bashrc
# enable color support of ls and also add handy aliases
    alias ls='ls --color=auto'
    #alias dir='dir --color=
    #alias vdir='vdir --color=
    #alias grep='grep --color=
    #alias fgrep='fgrep --color'
    #alias egrep='egrep --color=
# some more ls aliases
#ll='ls -al'
#alias la='ls -A'
#alias l='ls -CF'
# ~/.bash_aliases, instead of adding them here directly.
if [ -f ~/.bash_aliases ]; then
   . ~/.bash_aliases
```

Як ви можете прочитати в останніх трьох рядках, нам пропонується можливість мати власний файл, призначений для псевдонімів, `~/.bash_aliases`, і мати його вихідний код `.bashrc` під час кожного запуску системи. Отже, ми можемо вибрати цей варіант і створити та заповнити такий файл:

```
############
# .bash_aliases:
# a file to be populated by the user's personal aliases (and sourced by ~/.bashrc).
############
alias git_info='which git;git --version'
alias greet='echo Hello world!'
alias ll='ls -al'
```

```
alias where?='echo $PWD'
```

Створення функцій

Порівняно з псевдонімами, функції є більш програмними та гнучкими, особливо коли йдеться про використання повного потенціалу спеціальних вбудованих змінних і позиційних параметрів *Bash*. Вони також чудово підходять для роботи зі структурами керування потоком, такими як цикли чи умови. Ми можемо думати про функцію як про команду, яка включає логіку через блоки або колекції інших команд.

Два синтаксиси для створення функцій

Існує два дійсні синтаксиси для визначення функцій.

Використання ключового слова `function`

З одного боку, ми можемо використати ключове слово `function`, за яким йде назва функції та команди у фігурних дужках:

```
function function_name {
command #1
command #2
command #3
.
.
.
command #n
}
```

Використання `()`

З іншого боку, ми можемо виключити ключове слово `function` і використовувати замість нього дві дужки відразу після назви функції:

```
function_name() {
command #1
command #2
command #3
.
.
.
command #n
```

```
}
```

Звичайним явищем є розміщення функцій у файлах або скриптах. Однак їх також можна записати безпосередньо в підказку оболонки з кожною командою в окремому рядку - зверніть увагу на `PS2(>)`, що вказує на новий рядок після розриву рядка:

```
$ greet() {
> greeting="Hello world!"
> echo $greeting
> }
```

У будь-якому випадку, і незалежно від вибраного нами синтаксису, якщо ми вирішимо пропустити розриви рядків і написати функцію лише в одному рядку, команди мають бути розділені крапкою з комою (також зверніть увагу на крапку з комою після останньої команди):

```
$ greet() { greeting="Hello world!"; echo $greeting; }
```

`bash` не повідомляв про помилку, коли ми натискали Enter, тому наша функція готова до виклику. Щоб викликати функцію, ми повинні ввести її назву в термінал:

```
$ greet
Hello world!
```

Як і у випадку зі змінними та псевдонімами, якщо ми хочемо, щоб функції залишалися постійними під час перезавантаження системи, ми маємо помістити їх у сценарії ініціалізації оболонки, такі як `/etc/bash.bashrc` (глобальний) або `~/.bashrc` (локальний).

WARNING

Після додавання псевдонімів або функцій до будь-якого файлу сценарію запуску ви повинні вказати такі файли як `.` або `source`, щоб зміни набули чинності, якщо ви не хочете виходити з системи та повертатися знову або перезавантажувати систему.

Спеціальні вбудовані змінні Bash

Bourne Again Shell поставляється з набором спеціальних змінних, які особливо корисні для функцій і сценаріїв. Вони особливі, оскільки на них можна лише посилатися, а не призначати. Ось список найбільш актуальних з них:

`$?`

Посилання на цю змінну розширюється до результату виконання останньої команди. Значення `0` означає успішне виконання:

```
$ ps aux | grep bash
user2      420  0.0  0.4  21156  5012 pts/0    Ss   17:10   0:00 -bash
user2      640  0.0  0.0  12784   936 pts/0    S+   18:04   0:00 grep bash
$ echo $?
0
```

Значення, відмінне від `0`, означає помилку:

```
user1@debian:~$ ps aux |rep bash
-bash: rep: command not found
user1@debian:~$ echo $?
127
```

`$$`

Розширюється до PID оболонки (ID процесу):

```
$ ps aux | grep bash
user2      420  0.0  0.4  21156  5012 pts/0    Ss   17:10   0:00 -bash
user2      640  0.0  0.0  12784   936 pts/0    S+   18:04   0:00 grep bash
$ echo $$
420
```

`$!`

Розширюється до PID останнього фонового завдання:

```
$ ps aux | grep bash &
[1] 663
$ user2      420  0.0  0.4  21156  5012 pts/0    Ss+  17:10   0:00 -bash
user2      663  0.0  0.0  12784   972 pts/0    S    18:08   0:00 grep bash
^C
[1]+  Done                    ps aux | grep bash
$ echo $!
663
```

NOTE Пам’ятайте, що амперсанд (&) використовується для запуску процесів у

фоновому режимі.

Позиційні параметри від `$0` до `$9`

Розширюються до параметрів або аргументів, які передаються функції (псевдоніму чи сценарію) — `$0` розширюється до назви сценарію або оболонки.

Давайте створимо функцію для демонстрації позиційних параметрів. Зверніть увагу на `PS2` (`>`), що вказує на нові рядки після розривів рядків:

```
$ special_vars() {
> echo $0
> echo $1
> echo $2
> echo $3
}
```

Тепер ми викличемо функцію (`special_vars`), передавши їй три параметри (`debian`, `ubuntu`, `zorin`):

```
$ special_vars debian ubuntu zorin
-bash
debian
ubuntu
zorin
```

Все спрацювало, як і очікувалося.

WARNING

Хоча передача позиційних параметрів до псевдонімів технічно можлива, це зовсім не функціонально, оскільки, з псевдонімами, позиційні параметри завжди передаються в кінці:

```
$ alias great_editor='echo $1 is a great text editor'
$ great_editor emacs
is a great text editor emacs
```

Інші спеціальні вбудовані змінні Bash включають:

`$#`

Розширюється до кількості аргументів, переданих команді.

`$@`, `$*`

Розширюються до аргументів, переданих команді.

`$_`

Розширюється до останнього параметра або назви сценарію (серед іншого; див. `man bash`, щоб дізнатися більше!):

Змінні у функціях

Звичайно, змінні можна використовувати у функціях.

Щоб підтвердити це, цього разу ми створимо новий порожній файл під назвою `funed` і додамо до нього таку функцію:

```
editors() {

editor=emacs

echo "My editor is: $editor. $editor is a fun text editor."
}
```

Як ви вже могли здогадатися, ми повинні спочатку отримати файл, щоб мати можливість викликати функцію:

```
$ . funed
```

І тепер ми можемо це перевірити:

```
$ editors
My editor is emacs. emacs is a fun text editor.
```

Як ви розумієте, щоб функція `editors` працювала належним чином, спочатку потрібно встановити змінну `editor`. Область цієї змінної є локальною для поточної оболонки, і ми можемо посилатися на неї поки триває сеанс:

```
$ echo $editor
emacs
```

Разом із локальними змінними ми також можемо включити змінні середовища в нашу

функцію:

```
editors() {

editor=emacs

echo "The text editor of $USER is: $editor."
}

editors
```

Зверніть увагу, що цього разу ми вирішили викликати функцію з самого файлу (`editors` в останньому рядку). Таким чином, коли ми будемо створювати файл, функція також буде викликана - все одночасно:

```
$ . funed
The text editor of user2 is: emacs.
```

Позиційні параметри у функціях

Щось подібне відбувається з позиційними параметрами.

Ми можемо передати їх у функції з файлу чи сценарію (зверніть увагу на останній рядок: `editors tortoise`):

```
editors() {

editor=emacs

echo "The text editor of $USER is: $editor."
echo "Bash is not a $1 shell."
}

editors tortoise
```

Ми отримуємо файл і бачимо, що він працює:

```
$ . funed
The text editor of user2 is: emacs.
Bash is not a tortoise shell.
```

І ми також можемо передавати позиційні параметри функціям у командному рядку. Щоб підтвердити це, ми позбавляємося від останнього рядка файлу:

```
editors() {

editor=emacs

echo "The text editor of $USER is: $editor."
echo "Bash is not a $1 shell."
}
```

Потім ми повинні створити джерело файлу:

```
$ . funed
```

Нарешті, ми викликаємо функцію з `tortoise` як позиційний параметр `$1` у командному рядку:

```
$ editors tortoise
The text editor of user2 is: emacs.
Bash is not a tortoise shell.
```

Функції в сценаріях

Функції здебільшого містяться в сценаріях Bash.

Перетворення нашого файлу `funed` на сценарій (ми назвемо його `funed.sh`) — це справді надзвичайно:

```
#!/bin/bash

editors() {

editor=emacs

echo "The text editor of $USER is: $editor."
echo "Bash is not a $1 shell."
}

editors tortoise
```

Це все! Ми додали лише два рядки:

- Перший рядок — це *shebang*, він визначає, яка програма збирається інтерпретувати сценарій: `#!/bin/bash`. Цікаво, що ця програма сама по собі є bash.
- Останній рядок є просто викликом функції.

Тепер залишається тільки одне — ми повинні зробити скрипт виконуваним:

```
$ chmod +x funed.sh
```

І тепер він готовий до виконання:

```
$ ./funed.sh
The text editor of user2 is: emacs.
Bash is not a tortoise shell.
```

NOTE Ви дізнаєтеся все про *сценарії оболонки* у наступних кількох уроках.

Функція в псевдонімі

Як було сказано вище, ми можемо розмістити функцію в псевдонімі:

```
$ alias great_editor='gr8_ed() { echo $1 is a great text editor; unset -f gr8_ed; }; gr8_ed'
```

Цей довгий псевдонім заслуговує на пояснення. Давайте розберемо його:

- Спочатку сама функція: `gr8_ed() { echo $1 is a great text editor; unset -f gr8_ed; }`
- Остання команда у функції, `unset -f gr8_ed`, скасовує функцію, щоб вона не залишалася в поточному сеансі `bash` після виклику псевдоніма.
- І останнє, але не менш важливе: щоб мати успішний виклик псевдоніма, ми також повинні спочатку викликати функцію: `gr8_ed`.

Давайте викличемо псевдонім і перевіримо, що він працює:

```
$ great_editor emacs
emacs is a great text editor
```

Як показано в `unset -f gr8_ed` вище, команда `unset` використовується не лише для скасування змінних, але й для функцій. Насправді існують певні перемикачі або опції:

`unset -v`
: для змінних

`unset -f`
: для функцій

Якщо не використовувати перемикачі, `unset` спочатку спробує скасувати змінну, а якщо це не вдасться, спробує скасувати функцію.

Функція у функції

Тепер, наприклад, ми хочемо повідомляти `user2` про дві речі щоразу, коли він входить до системи:

- Привітатися та порекомендувати/похвалити текстовий редактор.
- Якщо він починає розміщувати багато відеофайлів `Matroska` у своїй папці `$HOME/Video`, ми також хочемо його про це попередити.

Щоб досягти цієї мети, ми розмістили такі дві функції в `/home/user2/.bashrc`:

Перша функція (`check_vids`) виконує перевірку файлів `.mkv` і надсилає попередження:

```
check_vids() {
    ls -1 ~/Video/*.mkv > /dev/null 2>&1
    if [ "$?" = "0" ];then
        echo -e "Remember, you must not keep more than 5 video files in your Video
folder.\nThanks."
    else
    echo -e "You do not have any videos in the Video folder. You can keep up to 5.\nThanks."
    fi
}
```

`check_vids` робить три речі:

- Вона виводить список файлів `mkv` у `~/Video`, надсилаючи вихідні дані і будь-які помилки до так званого *bit-bucket* (`/dev/null`).
- Вона перевіряє результат попередньої команди на успішність.
- Залежно від результату тесту, вона виводить одне з двох повідомлень.

Друга функція є модифікованою версією нашої функції `editors`:

```
editors() {

editor=emacs

echo "Hi, $USER!"
echo "$editor is more than a text editor!"

check_vids
}

editors
```

Важливо звернути увагу на дві речі:

- Остання команда `editors` викликає `check_vids`, тому обидві функції стають ланцюжком: привітання, рекомендація, перевірка та попередження виконуються послідовно.
- Сама функція `editors` є точкою входу до послідовності функцій, тому вона викликається в останньому рядку (`editors`).

Тепер давайте увійдемо як `user2` і перевіримо, що це працює:

```
# su - user2
Hi, user2!
emacs is more than a text editor!
Remember, you must not keep more than 5 video files in your Video folder.
Thanks.
```

Вправи до посібника

1. Заповніть таблицю “Так” або “Ні” з урахуванням можливостей псевдонімів і функцій:

Властивість	**Псевдонім?**	**Функція?**
Можна використовувати локальні змінні		
Можна використовувати змінні середовища		
Можна екранувати за допомогою `\`		
Може бути рекурсивним		
Висока ефективність при використанні з позиційними параметрами		

2. Введіть команду, яка виведе список усіх псевдонімів у вашій системі:

3. Напишіть псевдонім з назвою `logg`, який містить список усіх `ogg` файлів у `~/Music` - по одному на рядок:

4. Викличте псевдонім, щоб довести, що він працює:

5. Тепер змініть псевдонім так, щоб він виводив (за допомогою echo) користувача сеансу та двокрапку перед списком:

6. Викличте його знову, щоб підтвердити, що ця нова версія також працює:

7. Знову виведіть всі псевдоніми та переконайтеся, що ваш псевдонім `logg` відображається в списку:

8. Видаліть псевдонім:

9. Розгляньте стовпці “Назва псевдоніма” і “Команда(и) псевдоніма” і правильно призначте псевдонімам їх значення:

Псевдонім	**Команда(и) псевдоніма**	**Присвоєння псевдонімів**
`b`	`bash`	
`bash_info`	`which bash` + `echo "$BASH_VERSION"`	
`kernel_info`	`uname -r`	
`greet`	`echo Hi, $USER!`	
`computer`	`pc=slimbook` + `echo My computer is a $pc`	

10. Як `root`, напишіть функцію під назвою `my_fun` в `/etc/bash.bashrc`. Функція повинна передати користувачам привіт і повідомити їм, який їхній шлях. Викличте їх так, щоб користувач отримував обидва повідомлення кожного разу при вході:

11. Увійдіть як `user2`, щоб перевірити, чи функція працює:

12. Напишіть ту саму функцію в одному рядку:

13. Викличте функцію:

14. Видаліть функцію:

15. Це модифікована версія функції `special_vars`:

```
$ special_vars2() {
> echo $#
> echo $_
> echo $1
> echo $4
> echo $6
> echo $7
```

```
> echo $_
> echo $@
> echo $?
> }
```

Це команда, яку ми використовуємо для її виклику:

```
$ special_vars2 crying cockles and mussels alive alive oh
```

Вгадайте яким буде результат

Посилання	Значення
`echo $#`	
`echo $_`	
`echo $1`	
`echo $4`	
`echo $6`	
`echo $7`	
`echo $_`	
`echo $@`	
`echo $?`	

16. На основі зразка функції (`check_vids`) у розділі “Функція у функції”, напишіть функцію під назвою `check_music`, щоб включити її до сценарію запуску `bash`, який приймає позиційні параметри, щоб ми могли легко змінювати:
    - тип файлу, який перевіряється: `ogg`
    - каталог, у якому зберігаються файли: `~/Music`
    - тип файлу, який зберігається: `music`
    - кількість файлів, що зберігаються: `7`

Дослідницькі вправи

1. Функції лише для читання – це ті, вміст яких ми не можемо змінити. Проведіть дослідження щодо *функцій лише для читання* та заповніть наступну таблицю:

Назва функції	**Зробити лише для читання**	**Вивести всі функції лише для читання**
`my_fun`		

2. Знайдіть в Інтернеті, як змінити `PS1` та будь-що інше, що вам може знадобитися, щоб написати функцію під назвою `fyi` (для розміщення в сценарії запуску), яка надає користувачеві таку інформацію:
   - ім’я користувача
   - домашній каталог
   - ім’я власника
   - тип операційної системи
   - шлях пошуку виконуваних файлів
   - поштовий каталог
   - як часто перевіряється пошта
   - глибина поточного сеансу оболонки
   - підказка (ви повинні змінити її так, щоб вона відображала `<user>@<host-date>`)

Підсумки

В цьому уроці ми вивчили:

- Псевдоніми і функції є важливими функціями оболонки, які дозволяють нам інкапсулювати блоки коду, що повторюються.
- Псевдоніми корисні для створення коротших версій довгих та/або складних команд.
- Функції — це процедури, які реалізують логіку та дозволяють нам автоматизувати завдання, особливо коли вони використовуються в сценаріях.
- Синтаксис запису псевдонімів і функцій.
- Як об'єднати різні команди за допомогою крапки з комою (`;`).
- Як правильно використовувати лапки з псевдонімами.
- Як зробити псевдоніми та функції постійними.
- Спеціальні вбудовані змінні Bash: `$?`, `$$`, `$!`, позиційні параметри (`$0`-`$9`), `$#`, `$@`, `$*` і "`$_`".
- Як використовувати змінні та позиційні параметри з функціями.
- Як використовувати функції в сценаріях.
- Як викликати функцію з псевдоніма.
- Як викликати функцію з іншої функції.
- Основи створення сценарію `bash`.

Команди та ключові слова, які використовуються в цьому уроці:

`alias`

Створює псевдоніми.

`unalias`

Видаляє псевдоніми.

`cd`

Змінює каталог.

`grep`

Виводить рядки, що відповідають шаблону.

`function`

Ключове слово оболонки для створення функцій.

`.`

Джерело файлу.

`source`

Джерело файлу.

`ps`

Виводить перелік поточних процесів.

`echo`

Виводить рядок тексту.

`chmod`

Змінює біти властивості файлу, наприклад робить його виконуваним.

`unset`

Видаляє змінні і функції.

`su`

Змінює ідентифікатор користувача або встановлює суперкористувача.

Відповіді до вправ посібника

1. Заповніть таблицю “Так” або “Ні” з урахуванням можливостей псевдонімів і функцій:

Властивість	**Псевдонім?**	**Функція?**
Можна використовувати локальні змінні	Так	Так
Можна використовувати змінні середовища	Так	Так
Можна екранувати за допомогою `\`	Так	Ні
Може бути рекурсивним	Так	Так
Висока ефективність при використанні з позиційними параметрами	Ні	Так

2. Введіть команду, яка виведе список усіх псевдонімів у вашій системі:

```
alias
```

3. Напишіть псевдонім з назвою `logg`, який містить список усіх `ogg` файлів у `~/Music` - по одному на рядок:

```
alias logg='ls -1 ~/Music/*ogg'
```

4. Викличте псевдонім, щоб довести, що він працює:

```
logg
```

5. Тепер змініть псевдонім так, щоб він виводив (за допомогою echo) користувача сеансу та двокрапку перед списком:

```
alias logg='echo $USER:; ls -1 ~/Music/*ogg'
```

6. Викличте його знову, щоб підтвердити, що ця нова версія також працює:

```
logg
```

7. Знову виведіть всі псевдоніми та переконайтеся, що ваш псевдонім `logg` відображається в списку:

```
alias
```

8. Видаліть псевдонім:

```
unalias logg
```

9. Розгляньте стовпці “Назва псевдоніма” і “Команда(и) псевдоніма” і правильно призначте псевдонімам їх значення:

Псевдонім	Команда(и) псевдоніма	Присвоєння псевдонімів
`b`	`bash`	`alias b=bash`
`bash_info`	`which bash` + `echo "$BASH_VERSION"`	`alias bash_info='which bash; echo "$BASH_VERSION"'`
`kernel_info`	`uname -r`	`alias kernel_info='uname -r'`
`greet`	`echo Hi, $USER!`	`alias greet='echo Hi, $USER'`
`computer`	`pc=slimbook` + `echo My computer is a $pc`	`alias computer='pc=slimbook; echo My computer is a $pc'`

NOTE Одинарні лапки також можна замінити подвійними.

10. Як `root`, напишіть функцію під назвою `my_fun` в `/etc/bash.bashrc`. Функція повинна передати користувачам привіт і повідомити їм, який їхній шлях. Викличте її так, щоб користувач отримував обидва повідомлення кожного разу при вході:

Варіант A:

```
my_fun() {
```

```
echo Hello, $USER!
echo Your path is: $PATH
}
my_fun
```

Варіант B:

```
function my_fun {
echo Hello, $USER!
echo Your path is: $PATH
}
my_fun
```

11. Увійдіть як `user2`, щоб перевірити, чи функція працює:

```
su - user2
```

12. Напишіть ту саму функцію в одному рядку:

Варіант A:

```
my_fun() { echo "Hello, $USER!"; echo "Your path is: $PATH"; }
```

Варіант B:

```
function my_fun { echo "Hello, $USER!"; echo "Your path is: $PATH"; }
```

13. Викличте функцію:

```
my_fun
```

14. Видаліть функцію:

```
unset -f my_fun
```

15. Це модифікована версія функції `special_vars`:

```
$ special_vars2() {
> echo $#
> echo $_
> echo $1
> echo $4
> echo $6
> echo $7
> echo $_
> echo $@
> echo $?
> }
```

Це команда, яку ми використовуємо для її виклику:

```
$ special_vars2 crying cockles and mussels alive alive oh
```

Вгадайте яким буде результат:

Посилання	**Значення**
`echo $#`	`7`
`echo $_`	`7`
`echo $1`	`crying`
`echo $4`	`mussels`
`echo $6`	`alive`
`echo $7`	`oh`
`echo $_`	`oh`
`echo $@`	`crying cockles and mussels alive alive oh`
`echo $?`	`0`

16. На основі зразка функції (`check_vids`) у розділі “Функція у функції”, напишіть функцію під назвою `check_music`, щоб включити її до сценарію запуску `bash`, який приймає позиційні параметри, щоб ми могли легко змінювати:
    - тип файлу, який перевіряється: `ogg`
    - каталог, у якому зберігаються файли: `~/Music`

- тип файлу, який зберігається: `music`
- кількість файлів, що зберігаються: 7

```
check_music() {
    ls -1 ~/$1/*.$2 > ~/.mkv.log 2>&1
    if [ "$?" = "0" ];then
        echo -e "Remember, you must not keep more than $3 $4 files in your $1
folder.\nThanks."
    else
     echo -e "You do not have any $4 files in the $1 folder. You can keep up to
$3.\nThanks."
    fi
}

check_music Music ogg 7 music
```

Відповіді до дослідницьких вправ

1. Функції лише для читання – це ті, вміст яких ми не можемо змінити. Проведіть дослідження щодо *функцій лише для читання* та заповніть наступну таблицю:

Назва функції	**Зробити лише для читання**	**Вивести всі функції лише для читання**
`my_fun`	`readonly -f my_fun`	`readonly -f`

2. Знайдіть в Інтернеті, як змінити `PS1` та будь-що інше, що вам може знадобитися, щоб написати функцію під назвою `fyi` (для розміщення в сценарії запуску), яка надає користувачеві таку інформацію:
   - ім'я користувача
   - домашній каталог
   - ім'я власника
   - тип операційної системи
   - шлях пошуку виконуваних файлів
   - поштовий каталог
   - як часто перевіряється пошта
   - глибина поточного сеансу оболонки
   - підказка (ви повинні змінити її так, щоб вона відображала `<user>@<host-date>`)

```
fyi() {
    echo -e "For your Information:\n
    Username: $USER
    Home directory: $HOME
    Host: $HOSTNAME
    Operating System: $OSTYPE
    Path for executable files: $PATH
    Your mail directory is $MAIL and is searched every $MAILCHECK seconds.
    The current level of your shell is: $SHLVL"
    PS1="\u@\h-\d "
}

fyi
```

105.2 Налаштування або написання простих сценаріїв

Посилання на вимоги LPI

LPIC-1 5.0, Exam 102, Objective 105.2

Ваговий коефіцієнт

4

Ключові галузі знань

- Використання стандартного синтаксису sh (цикли, тести).
- Використання команд заміни.
- Виконання перевірки, що повертає значення успішного виконання чи помилки, або іншу інформацію, надану командою.
- Виконання ланцюгових команд.
- Виконання умовної розсилки суперкористувачу.
- Правильне визначення інтерпретатору сценарію через рядок shebang (#!).
- Керування розташуванням, правом власності, виконанням і правами suid для сценарію.

Частковий список файлів, термінів та утиліт, що використовуються

- `for`
- `while`
- `test`
- `if`
- `read`
- `seq`

- `exec`
- `||`
- `&&`

105.2 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	105 Оболонки та сценарії оболонки
Тема:	105.2 Налаштування або написання простих сценаріїв
Урок:	1 з 2

Вступ

Середовище оболонки Linux дозволяє використовувати файли, які називаються *сценарії*, що містять команди з будь-якої доступної програми в системі в поєднанні з вбудованими командами оболонки для автоматизації завдань користувача та/або системи. Дійсно, багато завдань обслуговування операційної системи виконуються сценаріями, що складаються з послідовностей команд, структур прийняття рішень та циклів з умовами. Хоча сценарії здебільшого призначені для завдань, пов'язаних із самою операційною системою, вони також корисні для орієнтованих на користувача завдань, таких як масове перейменування файлів, збір і синтаксичний аналіз даних або будь-які інші дії командного рядка, що повторюються.

Сценарії — це не що інше, як текстові файли, які поводяться як програми. Актуальна програма, інтерпретатор, читає та виконує інструкції, перелічені у файлі сценарію. Інтерпретатор також може розпочати інтерактивний сеанс, у якому команди, включаючи сценарії, зчитуються та виконуються під час їх введення, як у випадку з сеансами оболонки Linux. Файли сценаріїв можуть групувати ці інструкції та команди, коли вони стають надто складними, щоб їх можна було реалізувати як псевдонім або спеціальну

функцію оболонки. Крім того, файли сценаріїв можна підтримувати як звичайні програми, і, оскільки це просто текстові файли, їх можна створювати та змінювати за допомогою будь-якого простого текстового редактора.

Структура та виконання сценарію

По суті, файл сценарію — це впорядкована послідовність команд, які мають виконуватися відповідним інтерпретатором команд. Те, як інтерпретатор читає файл сценарію, різниться, і існують різні способи зробити це в сеансі оболонки Bash, але інтерпретатором за замовчуванням для файлу сценарію буде той, що вказано в першому рядку сценарію відразу після символів `\# !`(відомий як *shebang*). У сценарії з інструкціями для оболонки Bash перший рядок має бути `!/bin/bash`. Якщо вказати цей рядок, інтерпретатор для всіх інструкцій у файлі буде інтерпретатором /bin/bash. За винятком першого рядка, усі інші рядки, що починаються з решетки, ігноруватимуться, тому їх можна використовувати для розміщення нагадувань і коментарів. Порожні рядки також ігноруються. Тому дуже лаконічний файл сценарію оболонки можна записати так:

```
#!/bin/bash

# A very simple script

echo "Cheers from the script file! Current time is: "

date +%H:%M
```

Цей сценарій містить лише дві інструкції для інтерпретатора `/bin/bash`: вбудовану команду `echo` і команду `date`. Найпростіший спосіб запустити файл сценарію — це запустити інтерпретатор із шляхом до сценарію в якості аргументу. Отже, якщо припустити, що попередній приклад було збережено у файлі сценарію під назвою `script.sh` у поточному каталозі, його буде прочитано та інтерпретовано Bash за допомогою такої команди:

```
$ bash script.sh
Cheers from the script file! Current time is:
10:57
```

Команда `echo` автоматично додасть новий рядок після відображення вмісту, але параметр `-n` відмінить цю поведінку. Таким чином, використання `echo -n` у сценарії змусить виведення обох команд відображатися в одному рядку:

```
$ bash script.sh
Cheers from the script file! Current time is: 10:57
```

Хоча це не обов'язково, суфікс `.sh` допомагає ідентифікувати сценарії оболонки під час виведення їх у списку та пошуку файлів.

TIP

Bash викличе будь-яку команду, зазначену після `#!` як інтерпретатор для файлу сценарію. Наприклад, може бути корисним використовувати shebang для інших мов сценаріїв, таких як *Python* (`#!/usr/bin/python`), *Perl* (`#!/usr/bin/perl`) або *awk* (`#!/usr/bin/awk`).

Якщо файл сценарію призначений для виконання іншими користувачами в системі, важливо перевірити, чи встановлено належні дозволи на читання. Команда `chmod o+r script.sh` надасть дозвіл на читання всім користувачам у системі, дозволяючи їм виконувати `script.sh`, вказавши шлях до файлу сценарію як аргумент команди `bash`. Крім того, у файлі сценарію може бути встановлено дозвіл на біт виконання, щоб файл можна було виконати як звичайну команду. Біт виконання активується у файлі сценарію командою `chmod`:

```
$ chmod +x script.sh
```

Якщо біт виконання ввімкнено, файл сценарію під назвою `script.sh` у поточному каталозі можна виконати безпосередньо за допомогою команди `./script.sh`. Сценарії, розміщені в каталозі, зазначеному у змінній середовища `PATH`, також будуть доступні без повного шляху.

WARNING

Сценарій, який виконує обмежені дії, може мати активований дозвіл SUID, тому звичайні користувачі також можуть запускати сценарій із привілеями root. У цьому випадку дуже важливо переконатися, що жоден користувач, крім root, не має дозволу на запис у файлі. В іншому випадку звичайний користувач може змінити файл для виконання довільних і потенційно шкідливих операцій.

Розміщення та відступи команд у файлах сценаріїв не надто жорсткі. Кожен рядок у сценарії оболонки виконуватиметься як звичайна команда оболонки в тій самій послідовності, що й рядок у файлі сценарію, і ті самі правила, що застосовуються до підказки оболонки, також застосовуються до кожного рядка сценарію окремо. Можна розмістити дві або більше команд в одному рядку, розділених крапкою з комою:

```
echo "Cheers from the script file! Current time is:" ; date +%H:%M
```

Хоча цей формат іноді може бути зручним, його використання необов'язкове, оскільки послідовні команди можна розміщувати по одній команді в рядку, і вони виконуватимуться так само, як вони були розділені крапкою з комою. Іншими словами, крапку з комою можна замінити символом нового рядка у файлах сценарію Bash.

Коли сценарій виконується, команди, що містяться в ньому, не виконуються безпосередньо в поточному сеансі, а замість цього вони виконуються новим процесом Bash, який називається *sub-shell*. Це запобігає перезапису сценарієм змінних середовища поточного сеансу та залишенню неконтрольованих змін у поточному сеансі. Якщо мета полягає в тому, щоб запустити вміст сценарію в поточному сеансі оболонки, тоді його слід виконати за допомогою `source script.sh` або `. script.sh` (зверніть увагу, що між крапкою та назвою сценарію є пробіл).

Як це буває під час виконання будь-якої іншої команди, підказка оболонки знову буде доступною лише тоді, коли сценарій завершить своє виконання, а його код статусу виходу буде доступний у змінній `$?`. Щоб змінити цю поведінку, щоб поточна оболонка також завершувалася, коли завершується сценарій, сценарію або будь-якій іншій команді може передувати команда `exec`. Ця команда також замінить код статусу виходу поточного сеансу оболонки на власний.

Змінні

Змінні в сценаріях оболонки поводяться так само, як і в інтерактивних сеансах, якщо інтерпретатор той самий. Наприклад, формат `SOLUTION=42` (без пробілів навколо знака рівності) призначить значення `42` змінній з назвою `SOLUTION`. Згідно з домовленістю, для імен змінних використовуються великі літери, але це не є обов'язковим. Однак назви змінних не можуть починатися з неалфавітних символів.

На додаток до звичайних змінних, створених користувачем, сценарії Bash також мають набір спеціальних змінних, які називаються *параметри*. На відміну від звичайних змінних, імена параметрів починаються з неалфавітного символу, який позначає його функцію. Аргументи, передані сценарію, та інша корисна інформація зберігаються в таких параметрах, як `$0`, `$*`, `$?` тощо, де символ після знака долара вказує на інформацію, яку потрібно отримати:

`$*`

Всі аргументи передані в скрипт.

`$@`

Всі аргументи передані в скрипт. Якщо використовуються подвійні лапки, наприклад, `"$@"`, кожен аргумент буде взято в подвійні лапки.

`$#`

Кількість аргументів.

`$0`

Ім'я файлу сценарію.

`$!`

PID останньої виконаної програми.

`$$`

PID поточної оболонки.

`$?`

Числовий код статусу виходу останньої виконаної команди. Для стандартних процесів POSIX числове значення `0` означає, що остання команда була успішно виконана. Це також стосується сценаріїв оболонки.

Позиційний параметр — це параметр, позначений однією або декількома цифрами, окрім однієї цифри `0`. Наприклад, змінна `$1` відповідає першому аргументу, наданому скрипту (позиційний параметр один), `$2` відповідає другому аргументу і так далі. Якщо позиція параметра більша за дев'ять, на нього потрібно посилатися у фігурних дужках, як у `${10}`, `${11}` тощо.

З іншого боку, звичайні змінні призначені для зберігання вставлених вручну значень або результатів, створених іншими командами. Команда `read`, наприклад, може бути використана всередині сценарію, щоб попросити користувача ввести дані під час виконання сценарію:

```
echo "Do you want to continue (y/n)?"
read ANSWER
```

Повернене значення буде збережено в змінній `ANSWER`. Якщо ім'я змінної не вказано, за умовчанням використовуватиметься ім'я змінної REPLY. Також можна використовувати команду `read` для читання кількох змінних одночасно:

```
echo "Type your first name and last name:"
```

```
read NAME SURNAME
```

У цьому випадку кожне слово, розділене пробілами, буде присвоєно змінним `NAME` та `SURNAME` відповідно. Якщо кількість заданих термінів перевищує кількість змінних, то терміни, що перевищують, будуть збережені в останній змінній. Сам `read` може відобразити повідомлення користувачеві за допомогою параметра `-p`, що робить команду `echo` зайвою в цьому випадку:

```
read -p "Type your first name and last name:" NAME SURNAME
```

Сценарії, що виконують системні завдання, часто потребують інформації, наданої іншими програмами. *Зворотне позначення* можна використовувати для збереження результату команди в змінній:

```
$ OS=`uname -o`
```

У цьому прикладі результат команди `uname -o` зберігатиметься в змінній `OS`. Ідентичний результат буде отримано за допомогою `$()`:

```
$ OS=$(uname -o)
```

Довжина змінної, тобто кількість символів, які вона містить, повертається шляхом додавання символу # перед назвою змінної. Ця функція, однак, вимагає використання синтаксису фігурних дужок для позначення змінної:

```
$ OS=$(uname -o)
$ echo $OS
GNU/Linux
$ echo ${#OS}
9
```

Bash також містить одновимірні масиви змінних, тому набір пов’язаних елементів можна зберігати з одним іменем змінної. Кожен елемент у масиві має числовий індекс, який потрібно використовувати для запису та читання значень у відповідному елементі. На відміну від звичайних змінних, масиви мають бути оголошені за допомогою вбудованої команди Bash `declare`. Наприклад, щоб оголосити змінну з назвою `SIZES` як масив:

```
$ declare -a SIZES
```

Масиви також можуть бути неявно оголошені, коли вони заповнюються з попередньо визначеного списку елементів, використовуючи нотацію в дужках:

```
$ SIZES=( 1048576 1073741824 )
```

У прикладі два великих цілих значення зберігаються в масиві SIZES. На елементи масиву потрібно посилатися за допомогою фігурних і квадратних дужок, інакше Bash не змінить або не відобразить елемент правильно. Оскільки індекси масиву починаються з 0, вміст першого елемента знаходиться в `${SIZES[0]}`, другого елемента в `${SIZES[1]}` і так далі:

```
$ echo ${SIZES[0]}
1048576
$ echo ${SIZES[1]}
1073741824
```

На відміну від читання, зміна вмісту елемента масиву виконується без фігурних дужок (наприклад, `SIZES[0]=1048576`). Як і зі звичайними змінними, довжина елемента в масиві повертається з символом решітки (наприклад, `${#SIZES[0]}` для довжини першого елемента в масиві `SIZES`). Загальна кількість елементів у масиві повертається, якщо `@` або `*` використовуються як індекс:

```
$ echo ${#SIZES[@]}
2
$ echo ${#SIZES[*]}
2
```

Масиви також можуть бути оголошені, використовуючи вихідні дані команди як початкові елементи за допомогою підстановки команд. У наступному прикладі показано, як створити масив Bash, елементи якого є підтримуваними файловими системами поточної системи:

```
$ FS=( $(cut -f 2 < /proc/filesystems) )
```

Команда `cut -f 2 < /proc/filesystems` відобразить усі файлові системи, які наразі підтримуються запущеним ядром (як зазначено у другому стовпчику файлу `/proc/filesystems`), тому масив `FS` тепер містить один елемент для кожної файлової

системи, що підтримується. Для ініціалізації масиву можна використовувати будь-які текстові дані, оскільки за замовчуванням будь-які слова, розділені символами *space*, *tab* або *newline*, стануть елементом масиву.

TIP

Bash розглядає кожен символ `$IFS` (*Роздільник полів введення*) змінної середовища як роздільник. Щоб змінити роздільник поля лише на символи нового рядка, наприклад, змінну IFS потрібно скинути за допомогою команди `IFS=$'\n'`.

Арифметичні вирази

Bash надає практичний метод виконання цілочисельних арифметичних операцій за допомогою вбудованої команди `expr`. Дві числові змінні, наприклад, `$VAL1` і `$VAL2`, можна додати разом за допомогою наступної команди:

```
$ SUM=`expr $VAL1 + $VAL2`
```

Отримане значення прикладу буде доступне у змінній `$SUM`. Команду `expr` можна замінити на `$(())`, тож попередній приклад можна переписати як `SUM=$(( $VAL1 + $VAL2 ))`. Вирази ступеня також дозволені з оператором подвійних зірочок, тому попереднє оголошення масиву `SIZES=( 1048576 1073741824)` можна переписати як `SIZES=( $((1024**2)) $((1024) **3)) )`.

Підстановку команд також можна використовувати в арифметичних виразах. Наприклад, файл `/proc/meminfo` містить детальну інформацію про системну пам'ять, включаючи кількість вільних байтів в ОЗП:

```
$ FREE=$(( 1000 * `sed -nre '2s/[^[:digit:]]//gp' < /proc/meminfo` ))
```

У прикладі показано, як команду `sed` можна використовувати для аналізу вмісту `/proc/meminfo` всередині арифметичного виразу. Другий рядок файлу `/proc/meminfo` містить обсяг вільної пам'яті в тисячах байтів, тому арифметичний вираз множить його на 1000, щоб отримати кількість вільних байтів в RAM.

Умовне виконання

Деякі сценарії зазвичай призначені не для виконання всіх команд у файлі сценарію, а лише для тих команд, які відповідають попередньо визначеним критеріям. Наприклад, сценарій обслуговування може надіслати попередження на електронну пошту

адміністратора, лише якщо виконати команду не вдається. Bash надає спеціальні методи оцінки успішності виконання команд і загальні умовні структури, більш схожі на ті, які є в популярних мовах програмування.

Розділивши команди за допомогою `&&`, команда праворуч буде виконана, лише якщо команда ліворуч не виявила помилки, тобто якщо її статус виходу дорівнював `0`:

```
COMMAND A && COMMAND B && COMMAND C
```

Протилежна поведінка відбувається, якщо команди розділені символом `||`. У цьому випадку наступна команда буде виконана лише в тому випадку, якщо попередня команда виявила помилку, тобто якщо її повернутий код статусу відрізняється від 0.

Однією з найважливіших особливостей усіх мов програмування є можливість виконання команд залежно від попередньо визначених умов. Найпростішим способом умовного виконання команд є використання вбудованої команди Bash `if`, яка виконує одну або кілька команд, лише якщо команда, задана як аргумент, повертає код статусу 0 (успішно). Інша команда, `test`, може бути використана для оцінки багатьох різних спеціальних критеріїв, тому вона здебільшого використовується в поєднанні з `if`. У наведеному нижче прикладі повідомлення `Confirmed: /bin/bash is executable.` буде показано, якщо файл `/bin/bash` існує і він виконуваний:

```
if test -x /bin/bash ; then
  echo "Confirmed: /bin/bash is executable."
fi
```

Опція `-x` змушує команду `test` повертати код статусу 0, лише якщо вказаний шлях є виконуваним файлом. У наступному прикладі показано інший спосіб досягнення точно такого ж результату, оскільки квадратні дужки можна використовувати як заміну для `test`:

```
if [ -x /bin/bash ] ; then
  echo "Confirmed: /bin/bash is executable."
fi
```

Інструкція `else` є необов'язковою для структури `if` і може, за наявності, визначати команду або послідовність команд для виконання, якщо умовний вираз не є істинним:

```
if [ -x /bin/bash ] ; then
```

```
  echo "Confirmed: /bin/bash is executable."
else
  echo "No, /bin/bash is not executable."
fi
```

Структури `if` мають завжди закінчуватися на `fi`, щоб інтерпретатор Bash знав, де закінчуються умовні команди.

Виведення сценарію

Навіть якщо мета сценарію включає лише операції, орієнтовані на файли, важливо відображати повідомлення, пов'язані з прогресом, у стандартному виведенні, щоб користувач був поінформований про будь-які проблеми та міг зрештою використовувати ці повідомлення для створення журналів операцій.

Вбудована команда Bash `echo` зазвичай використовується для відображення простих рядків тексту, але вона також надає деякі розширені функції. З опцією `-e` команда `echo` може відображати спеціальні символи за допомогою екранованих послідовностей (послідовність зворотної похилої риски, що позначає спеціальний символ). Наприклад:

```
#!/bin/bash

# Get the operating system's generic name
OS=$(uname -o)

# Get the amount of free memory in bytes
FREE=$(( 1000 * `sed -nre '2s/[^[:digit:]]//gp' < /proc/meminfo` ))

echo -e "Operating system:\t$OS"
echo -e "Unallocated RAM:\t$(( $FREE / 1024**2 )) MB"
```

Хоча використання лапок є необов'язковим при використанні `echo` без параметрів, необхідно додати їх при використанні параметра `-e`, інакше спеціальні символи можуть відтворюватися неправильно. У попередньому сценарії обидві команди `echo` використовують символ табуляції `\t` для вирівнювання тексту, що призводить до такого результату:

```
Operating system:       GNU/Linux
Unallocated RAM:        1491 MB
```

Символ нового рядка `\n` може бути використаний для розділення рядків виведення, тому точно таке виведення буде отримано шляхом поєднання двох команд `echo` в одну:

```
echo -e "Operating system:\t$OS\nUnallocated RAM:\t$(( $FREE / 1024**2 )) MB"
```

Хоча команда `echo` підходить для відображення більшості текстових повідомлень, вона може не підходити для відображення більш конкретних текстових шаблонів. Вбудована команда bash `printf` дає більше контролю над тим, як відображати змінні. Команда `printf` використовує перший аргумент як формат виведення, де заповнювачі будуть замінені наступними аргументами в тому порядку, в якому вони з'являються в командному рядку. Наприклад, повідомлення з попереднього прикладу можна створити за допомогою такої команди `printf`:

```
printf "Operating system:\t%s\nUnallocated RAM:\t%d MB\n" $OS $(( $FREE / 1024**2 ))
```

Заповнювач `%s` призначений для текстового вмісту (його буде замінено змінною `$OS`), а `%d` призначений для цілих чисел (його буде замінено результуючою кількістю вільних мегабайт в RAM). `printf` не додає символ нового рядка в кінці тексту, тому символ нового рядка `\n` слід розмістити в кінці шаблону, якщо це необхідно. Весь шаблон слід інтерпретувати як один аргумент, тому він повинен бути взятий у лапки.

TIP

Формат заміни заповнювача, який виконується `printf`, можна налаштувати за допомогою того самого формату, що використовується функцією `printf` з мови програмування C. Повну довідку про функцію `printf` можна знайти на її сторінці посібника, доступ до якої здійснюється за допомогою команди `man 3 printf`.

За допомогою `printf` змінні розміщуються поза текстовим шаблоном, що дає змогу зберігати текстовий шаблон в окремій змінній:

```
MSG='Operating system:\t%s\nUnallocated RAM:\t%d MB\n'
printf "$MSG" $OS $(( $FREE / 1024**2 ))
```

Цей метод особливо корисний для відображення різних вихідних форматів залежно від вимог користувача. Це полегшує, наприклад, написання сценарію, який використовує чіткий текстовий шаблон, якщо користувачеві потрібен список CSV (*значення, розділені комами*), а не стандартне вихідне повідомлення.

Вправи до посібника

1. Параметр `-s` команди `read` корисний для введення паролів, оскільки він не відображатиме вміст, який вводиться на екрані. Як можна використати команду `read` для збереження введених користувачем даних у змінній `PASSWORD`, приховуючи введений вміст?

2. Єдиною метою команди `whoami` є відображення імені користувача, який її викликав, тому вона здебільшого використовується всередині сценаріїв для ідентифікації користувача, який їх виконує. Як можна зберегти в сценарії Bash результат команди `whoami` у змінній з назвою `WHO`?

3. Який оператор Bash має бути між командами `apt-get dist-upgrade` і `systemctl reboot`, якщо користувач root хоче виконати `systemctl reboot`, лише якщо `apt-get dist-upgrade` завершено успішно?

Дослідницькі вправи

1. Після спроби запустити щойно створений сценарій Bash користувач отримує таке повідомлення про помилку:

   ```
   bash: ./script.sh: Permission denied
   ```

   Враховуючи, що файл `./script.sh` створено тим самим користувачем, яка ймовірна причина цієї помилки?

2. Припустимо, що файл сценарію з назвою `do.sh` є виконуваним і символічне посилання з назвою `undo.sh` вказує на нього. Як за допомогою сценарію визначити, яке ім'я файлу виклику було --`do.sh` чи `undo.sh`?

3. У системі з правильно налаштованою службою електронної пошти команда `mail -s "Maintenance Error" root <<<"Scheduled task error"` надсилає сповіщення електронною поштою користувачеві root. Таку команду можна використовувати в автоматичних завданнях, наприклад, *cronjobs*, щоб повідомити системного адміністратора про неочікувану проблему. Напишіть конструкцію *if*, яка виконає вищезгадану команду `mail`, якщо статус завершення попередньої команди, яким би він не був, є невдалим.

Підсумки

Цей урок охоплює основні поняття для розуміння та написання сценаріїв оболонки Bash. Сценарії оболонки є основною частиною будь-якого дистрибутива Linux, оскільки вони пропонують дуже гнучкий спосіб автоматизації завдань користувача та системи, що виконуються в середовищі оболонки. В уроці розглянуто наступні питання:

- Структура сценарію оболонки та правильні дозволи файлу сценарію
- Параметри сценарію
- Використання змінних для читання введених користувачем даних і збереження результатів команд
- Масиви Bash
- Прості тести та умовне виконання
- Форматування виведення

Розглянуті команди та процедури:

- Вбудована нотація Bash для підстановки команд, розширення масиву та арифметичних виразів
- Умовне виконання команд за допомогою операторів `||` і `&&`
- `echo`
- `chmod`
- `exec`
- `read`
- `declare`
- `test`
- `if`
- `printf`

Відповіді до вправ посібника

1. Параметр `-s` команди `read` корисний для введення паролів, оскільки він не відображатиме вміст, який вводиться на екрані. Як можна використати команду `read` для збереження введених користувачем даних у змінній `PASSWORD`, приховуючи введений вміст?

   `read -s PASSWORD`

2. Єдиною метою команди `whoami` є відображення імені користувача, який її викликав, тому вона здебільшого використовується всередині сценаріїв для ідентифікації користувача, який їх виконує. Як можна зберегти в сценарії Bash результат команди `whoami` у змінній з назвою `WHO`?

   `WHO=\`whoami\`` or `WHO=$(whoami)`

3. Який оператор Bash має бути між командами `apt-get dist-upgrade` і `systemctl reboot`, якщо користувач root хоче виконати `systemctl reboot`, лише якщо `apt-get dist-upgrade` завершено успішно?

   Оператор `&&`, як у `apt-get dist-upgrade && systemctl reboot`.

Відповіді до дослідницьких вправ

1. Після спроби запустити щойно створений сценарій Bash користувач отримує таке повідомлення про помилку:

   ```
   bash: ./script.sh: Permission denied
   ```

   Враховуючи, що файл `./script.sh` створено тим самим користувачем, яка ймовірна причина цієї помилки?

   Для файлу `./script.sh` не ввімкнено дозвіл на виконання.

2. Припустимо, що файл сценарію з назвою `do.sh` є виконуваним і символічне посилання з назвою `undo.sh` вказує на нього. Як за допомогою сценарію визначити, яке ім'я файлу виклику було `--do.sh` чи `undo.sh`?

   Спеціальна змінна `$0` містить назву файлу, який використовується для виклику сценарію.

3. У системі з правильно налаштованою службою електронної пошти команда `mail -s "Maintenance Error" root <<<"Scheduled task error"` надсилає сповіщення електронною поштою користувачеві root. Таку команду можна використовувати в автоматичних завданнях, як-от *cronjobs*, щоб повідомити системного адміністратора про неочікувану проблему. Напишіть конструкцію *if*, яка виконає вищезгадану команду `mail`, якщо статус завершення попередньої команди — яким би він не був — є невдалим.

   ```
   if [ "$?" -ne 0 ]; then mail -s "Maintenance Error" root <<<"Scheduled task error"; fi
   ```

105.2 Урок 2

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	105 Оболонки та сценарії оболонки
Тема:	105.2 Налаштування або написання простих сценаріїв
Урок:	2 з 2

Вступ

Сценарії оболонки зазвичай призначені для автоматизації операцій, пов’язаних з файлами та каталогами, тих самих операцій, які можна виконувати вручну в командному рядку. Проте сценарії оболонки не обмежуються лише документами користувача, оскільки конфігурація та взаємодія з багатьма аспектами операційної системи Linux також здійснюються за допомогою файлів сценаріїв.

Оболонка Bash пропонує багато корисних вбудованих команд для написання сценаріїв оболонки, але повна потужність цих сценаріїв залежить від поєднання вбудованих команд Bash із багатьма утилітами командного рядка, доступними в системі Linux.

Розширені тести

Bash як мова сценаріїв здебільшого орієнтована на роботу з файлами, тому вбудована команда Bash `test` має багато параметрів для оцінки властивостей об’єктів файлової системи (по суті, файлів і каталогів). Тести, зосереджені на файлах і каталогах, корисні, наприклад, для перевірки наявності файлів і каталогів, необхідних для виконання певного

завдання, а також для інформації щодо можливості їх читання. Потім, пов'язаний з умовною конструкцією *if*, правильний набір дій виконується, якщо тест проходить успішно.

Команда `test` може обчислювати вирази за допомогою двох різних синтаксисів: тестові вирази можна вказати як аргумент команди `test` або їх можна помістити в квадратні дужки, де команда `test` вказана неявно. Таким чином, тест для оцінки того, чи `/etc` є дійсним каталогом, можна записати як `test -d /etc` або як `[ -d /etc]`:

```
$ test -d /etc
$ echo $?
0
$ [ -d /etc ]
$ echo $?
0
```

Як підтверджено кодами статусу виходу в спеціальній змінній `$?` - значення 0 означає, що перевірка була успішною - обидві форми оцінили `/etc` як дійсний каталог. Якщо припустити, що шлях до файлу чи каталогу зберігається в змінній `$VAR`, такі вирази можна використовувати як аргументи `test` або в квадратних дужках:

`-a "$VAR"`

Визначає, чи існує шлях у `VAR` у файловій системі і чи є він файлом.

`-b "$VAR"`

Визначає, чи шлях у `VAR` є спеціальним файлом блоку.

`-c "$VAR"`

Визначає, чи шлях у `VAR` є файлом спеціальних символів.

`-d "$VAR"`

Визначає, чи шлях у `VAR` є каталогом.

`-e "$VAR"`

Визначає, чи існує шлях у `VAR` у файловій системі.

`-f "$VAR"`

Визначає, чи існує шлях у `VAR` і чи це звичайний файл.

`-g "$VAR"`

Визначає, чи має шлях у `VAR` дозвіл SGID.

`-h "$VAR"`

Визначає, чи шлях у `VAR` є символічним посиланням.

`-L "$VAR"`

Визначає, чи шлях у `VAR` є символічним посиланням (наприклад, `-h`).

`-k "$VAR"`

Визначає, чи має шлях у `VAR` дозвіл *sticky* bit.

`-p "$VAR"`

Визначає, чи шлях у `VAR` є файлом *pipe*.

`-r "$VAR"`

Визначає, чи шлях у `VAR` читається поточним користувачем.

`-s "$VAR"`

Визначає, чи існує шлях у `VAR` і чи він не порожній.

`-S "$VAR"`

Визначає, чи шлях у `VAR` є файлом сокета.

`-t "$VAR"`

Визначає, чи шлях у `VAR` відкритий у терміналі.

`-u "$VAR"`

Визначає, чи шлях у `VAR` має дозвіл SUID.

`-w "$VAR"`

Визначає, чи шлях у `VAR` доступний для запису поточним користувачем.

`-x "$VAR"`

Визначає, чи шлях у `VAR` є доступним для виконання поточним користувачем.

`-O "$VAR"`

Визначає, чи шлях у `VAR` належить поточному користувачеві.

`-G "$VAR"`

Визначає, чи шлях у `VAR` належить до ефективної групи поточного користувача.

`-N "$VAR"`

Визначає, чи шлях у `VAR` був змінений після останнього доступу до нього.

`"$VAR1" -nt "$VAR2"`

Визначає, чи шлях у `VAR1` є новішим за шлях у `VAR2`, відповідно до їх дат модифікації.

`"$VAR1" - від "$VAR2"`

Визначає, чи шлях у `VAR1` не старший за `VAR2`.

`"$VAR1" -ef "$VAR2"`

Цей вираз має значення True, якщо шлях у VAR1 є жорстким посиланням на VAR2.

Рекомендується використовувати подвійні лапки навколо змінної, яка перевіряється, тому що, якщо змінна порожня, це може спричинити синтаксичну помилку для команди `test`. Параметри перевірки вимагають аргументу операнда, а порожня змінна без лапок спричинить помилку через відсутність необхідного аргументу. Існують також тести для довільних текстових змінних, описані таким чином:

`-z "$TXT"`

Визначає, чи змінна `TXT` порожня (нульовий розмір).

`-n "$TXT"` або `test "$TXT"`

Визначає, чи змінна `TXT` не порожня.

`"$TXT1" = "$TXT2"` або `"$TXT1" == "$TXT2"`

Визначає, чи рівні `TXT1` і `TXT2`.

`"$TXT1" != "$TXT2"`

Визначає, чи не рівні `TXT1` і `TXT2`.

`"$TXT1" < "$TXT2"`

Визначає, чи стоїть `TXT1` перед `TXT2` в алфавітному порядку.

`"$TXT1" > "$TXT2"`

Визначає, чи стоїть `TXT1` після `TXT2` в алфавітному порядку.

Різні мови можуть мати різні правила алфавітного впорядкування. Щоб отримати узгоджені результати, незалежно від налаштувань локалізації системи, де виконується сценарій, рекомендується встановити змінну середовища `LANG` на `C`, як у `LANG=C`, перед виконанням операцій, що включають алфавітний порядок. Це визначення також зберігатиме системні повідомлення мовою оригіналу, тому його слід використовувати

лише в межах сценарію.

Числові порівняння мають власний набір параметрів перевірки:

`$NUM1 -lt $NUM2`
Визначає, чи значення `NUM1` менше за `NUM2`.

`$NUM1 -gt $NUM2`
Визначає, чи `NUM1` більше за `NUM2`.

`$NUM1 -le $NUM2`
Визначає, чи `NUM1` менше або дорівнює `NUM2`.

`$NUM1 -ge $NUM2`
Визначає, чи `NUM1` більше або дорівнює `NUM2`.

`$NUM1 -eq $NUM2`
Визначає, чи дорівнює `NUM1` `NUM2`.

`$NUM1 -не $NUM2`
Визначає, чи не дорівнює `NUM1` числу `NUM2`.

Усі тести можуть отримати такі модифікатори:

`! EXPR`
Визначає, чи є вираз `EXPR` хибним.

`EXPR1 -a EXPR2`
Визначає, чи істинні обидва значення: `EXPR1` і `EXPR2`.

`EXPR1 -o EXPR2`
Визначає, чи правдивий хоча б один із двох виразів.

Іншу умовну конструкцію, `case`, можна розглядати як варіацію конструкції *if*. Інструкція `case` виконає список наданих команд, якщо вказаний елемент, наприклад, вміст змінної, можна знайти в списку елементів, розділених *каналами* (вертикальна смуга `|`) і закінчених символом `)`. Наступний приклад сценарію показує, як можна використовувати конструкцію `case` для вказівки відповідного формату створення пакунків програмного забезпечення для даного дистрибутива Linux:

```
#!/bin/bash
```

```
DISTRO=$1

echo -n "Distribution $DISTRO uses "
case "$DISTRO" in
    debian | ubuntu | mint)
    echo -n "the DEB"
  ;;
    centos | fedora | opensuse )
    echo -n "the RPM"
  ;;
    *)
    echo -n "an unknown"
  ;;
esac
echo " package format."
```

Кожен список шаблонів і пов'язаних команд має закінчуватися символами `;;`, `;&` або `;;&`. Останній шаблон, позначений зірочкою, збігатиметься, якщо не було відповідності жодному іншому попередньому шаблону. Інструкція `esac` (*case* назад) завершує конструкцію `case`. Якщо припустити, що попередній приклад сценарію мав назву `script.sh` і він виконується з `opensuse` як першим аргументом, буде згенерована така вихідна інформація:

```
$ ./script.sh opensuse
Distribution opensuse uses the RPM package format.
```

TIP

Bash має опцію під назвою `nocasematch`, яка вмикає без урахування регістру відповідність шаблону для конструкції `case` та інших умовних команд. Вбудована команда `shopt` перемикає значення налаштувань, що контролюють необов'язкову поведінку оболонки: `shopt -s` увімкне (*set*) вказаний параметр, а `shopt -u` вимкне (*unset*) вказаний параметр. Таким чином, розміщення `shopt -s nocasematch` перед case-конструкцією увімкне регістронезалежне зіставлення шаблону. Параметри, змінені `shopt`, впливатимуть лише на поточний сеанс, тому змінені параметри всередині сценаріїв, що виконуються у суб-оболонці (що є стандартним способом запуску сценарію), не впливають на параметри батьківського сеансу.

Шуканий елемент і шаблони піддаються розширенню тильди, розширенню параметрів, заміні команд і арифметичному розширенню. Якщо шуканий елемент указано в лапках, їх буде видалено перед спробою зіставлення.

Цикли

Сценарії часто використовуються як інструмент для автоматизації завдань, що повторюються, виконуючи той самий набір команд, доки не буде перевірено критерій зупинки. Bash має три типи циклів — `for`, `until` і `while` — дещо різні циклічні конструкції.

Конструкція `for` проходить заданим списком елементів - зазвичай це список слів або будь-яких інших текстових сегментів, розділених пробілами - виконуючи той самий набір команд для кожного з цих елементів. Перед кожною ітерацією інструкція `for` призначає поточний елемент змінній, яка потім може використовуватися вкладеними командами. Процес повторюється доки не залишиться елементів. Синтаксис конструкції for:

```
for VARNAME in LIST
do
    COMMANDS
done
```

VARNAME – це довільне ім'я змінної оболонки, а LIST – будь-яка послідовність розділених елементів. Актуальні розмежувальні символи, що розділяють елементи в списку, визначаються змінною середовища `IFS`, якою за замовчуванням є символи *пробіл*, *табуляція* і *новий рядок*. Список команд, які потрібно виконати, розмежовано інструкціями `do` і `done`, тому команди можуть займати стільки рядків, скільки потрібно.

У наступному прикладі команда `for` візьме кожен елемент із наданого списку — послідовності чисел — і призначить його змінній `NUM`, по одному елементу за раз:

```
#!/bin/bash

for NUM in 1 1 2 3 5 8 13
do
    echo -n "$NUM is "
    if [ $(( $NUM % 2 )) -ne 0 ]
    then
        echo "odd."
    else
        echo "even."
  fi
done
```

У цьому прикладі вкладена конструкція `if` використовується в поєднанні з арифметичним

виразом, щоб визначити, чи є число в поточній змінній `NUM` парним чи непарним. Якщо припустити, що попередній зразок сценарію мав назву `script.sh` і він знаходиться в поточному каталозі, буде згенеровано такі вихідні дані:

```
$ ./script.sh
1 is odd.
1 is odd.
2 is even.
3 is odd.
5 is odd.
8 is even.
13 is odd.
```

Bash також підтримує альтернативний формат для конструкцій `for` із нотацією подвійних круглих дужок. Ця нотація нагадує синтаксис інструкцій `for` з мови програмування C і особливо корисна для роботи з масивами:

```
#!/bin/bash

SEQ=( 1 1 2 3 5 8 13 )

for (( IDX = 0; IDX < ${#SEQ[*]}; IDX++ ))
do
    echo -n "${SEQ[$IDX]} is "
    if [ $(( ${SEQ[$IDX]} % 2 )) -ne 0 ]
    then
        echo "odd."
    else
        echo "even."
  fi
done
```

Цей зразок сценарію генеруватиме той самий результат, що й попередній приклад. Однак замість використання змінної `NUM` для зберігання по одному елементу використовується змінна `IDX` для відстеження поточного індексу масиву в порядку зростання, починаючи з 0 і постійно збільшуючи його, поки існує номер елементу в масиві `SEQ`. Фактичний елемент отримується з його позиції в масиві за допомогою `${SEQ[$IDX]}`.

Так само конструкція `until` виконує послідовність команд, поки тестова команда, як і сама команда `test`, не закінчиться зі статусом 0 (успішно). Наприклад, ту саму структуру циклу з попереднього прикладу можна реалізувати за допомогою `until` наступним чином:

```
#!/bin/bash

SEQ=( 1 1 2 3 5 8 13 )

IDX=0

until [ $IDX -eq ${#SEQ[*]} ]
do
    echo -n "${SEQ[$IDX]} is "
    if [ $(( ${SEQ[$IDX]} % 2 )) -ne 0 ]
    then
        echo "odd."
    else
        echo "even."
  fi
  IDX=$(( $IDX + 1 ))
done
```

Конструкції з `until` можуть вимагати більше інструкцій, ніж конструкції з `for`, але вони можуть бути більш придатними для нечислових критеріїв зупинки, які надаються виразами `test` або будь-якою іншою командою. Важливо включати дії, які забезпечують дійсні критерії зупинки, наприклад, збільшення змінної лічильника, інакше цикл може працювати нескінченно.

Інструкція `while` подібна до інструкції `until`, але `while` продовжує повторювати набір команд, якщо тестова команда завершується зі статусом 0 (успішно). Таким чином, інструкція `until [ $IDX -eq ${#SEQ[*]} ]` з попереднього прикладу еквівалентна `while [ $IDX -lt ${#SEQ[*]} ]`, оскільки цикл має повторюватися, поки індекс масиву *менший за* загальну кількість елементів у масиві.

Більш докладний приклад

Уявіть собі, що користувач хоче періодично синхронізувати колекцію своїх файлів і каталогів з іншим запам'ятовуючим пристроєм, змонтованим у довільній точці монтування у файловій системі, і повнофункціональна система резервного копіювання вважається надмірною. Оскільки це дія, призначена для періодичного виконання, це хороший зразок для прикладу автоматизації за допомогою сценарію оболонки.

Завдання просте: синхронізувати кожен файл і каталог, що міститься у списку, від початкового каталогу, повідомленого як перший аргумент сценарію, до каталогу призначення, повідомленого як другий аргумент сценарію. Щоб було легше додавати або

видаляти елементи зі списку, він зберігатиметься в окремому файлі `~/.sync.list`, по одному елементу на рядок:

```
$ cat ~/.sync.list
Documents
To do
Work
Family Album
.config
.ssh
.bash_profile
.vimrc
```

Файл містить суміш файлів і каталогів, деякі з них мають пробіли в іменах. Це придатний сценарій для вбудованої команди Bash `mapfile`, яка аналізуватиме будь-який текстовий вміст і створюватиме з нього змінну масиву, розміщуючи кожен рядок як окремий елемент масиву. Файл сценарію матиме назву `sync.sh` і міститиме такий сценарій:

```
#!/bin/bash

set -ef

# List of items to sync
FILE=~/.sync.list

# Origin directory
FROM=$1

# Destination directory
TO=$2

# Check if both directories are valid
if [ ! -d "$FROM" -o ! -d "$TO" ]
then
  echo Usage:
  echo "$0 <SOURCEDIR> <DESTDIR>"
  exit 1
fi

# Create array from file
mapfile -t LIST < $FILE
```

```
# Sync items
for (( IDX = 0; IDX < ${#LIST[*]}; IDX++ ))
do
    echo -e "$FROM/${LIST[$IDX]} \u2192 $TO/${LIST[$IDX]}";
    rsync -qa --delete "$FROM/${LIST[$IDX]}" "$TO";
done
```

Перша дія, яку виконує сценарій, це перевизначення двох параметрів оболонки за допомогою команди `set`: параметр `-e` негайно припинить виконання, якщо команда виходить із ненульовим статусом, а параметр `-f` вимкне глоббінг імені файлу. Обидва варіанти можна скоротити як `-ef`. Це не обов'язковий крок, але він допомагає зменшити ймовірність неочікуваної поведінки.

Фактичні прикладні інструкції файлу сценарію можна розділити на три частини:

1. *Збирання і перевірка параметрів сценарію*

   Змінна `FILE` — це шлях до файлу, який містить список елементів, що потрібно скопіювати: `~/.sync.list`. Змінні `FROM` і `TO` є початковим і кінцевим шляхами відповідно. Оскільки останні два параметри надає користувач, вони проходять простий перевірочний тест, який виконується конструкцією `if`: якщо будь-який із двох не є дійсним каталогом, оцінюється за допомогою тестування `[ ! -d "$FROM" -o ! -d "$TO" ]` — цей сценарій покаже коротке довідкове повідомлення, а потім завершиться зі статусом виходу 1.

2. *Завантаження списку файлів і каталогів*

   Після визначення всіх параметрів за допомогою команди `mapfile -t LIST < $FILE` створюється масив, що містить список елементів, які потрібно скопіювати. Опція `-t` `mapfile` видалить кінцевий символ нового рядка з кожного рядка перед тим, як включити його до змінної масиву з назвою `LIST`. Вміст файлу, позначений змінною `FILE` — `~/.sync.list` — читається за допомогою перенаправлення введення.

3. *Виконання копіювання та повідомлення користувача*

   Цикл `for` із використанням нотації подвійних дужок проходить через масив елементів, а змінна `IDX` відстежує приріст індексу. Команда `echo` повідомлятиме користувача про кожен елемент, який копіюється. Екранований символ Юнікоду `\u2192` для символу *стрілки вправо* присутній у вихідному повідомленні, тому необхідно використовувати параметр `-e` команди `echo`. Команда `rsync` вибірково копіюватиме лише змінені частини файлу з джерела, тому її використання рекомендовано для таких завдань. Параметри `rsync` `-q` і `-a`, зведені до `-qa`, блокуватимуть повідомлення `rsync` і

активуватимуть режим *архівування*, у якому зберігаються всі властивості файлу. Параметр `--delete` змусить `rsync` видалити елемент у цільовій системі, який більше не існує в джерелі, тому його слід використовувати обережно.

Якщо припустити, що всі елементи в списку існують у домашньому каталозі користувача `carol`, `/home/carol`, а цільовий каталог `/media/carol/backup` вказує на підключений зовнішній пристрій зберігання даних, команда `sync. sh /home/carol /media/carol/backup` згенерує такий результат:

```
$ sync.sh /home/carol /media/carol/backup
/home/carol/Documents → /media/carol/backup/Documents
/home/carol/"To do" → /media/carol/backup/"To do"
/home/carol/Work → /media/carol/backup/Work
/home/carol/"Family Album" → /media/carol/backup/"Family Album"
/home/carol/.config → /media/carol/backup/.config
/home/carol/.ssh → /media/carol/backup/.ssh
/home/carol/.bash_profile → /media/carol/backup/.bash_profile
/home/carol/.vimrc → /media/carol/backup/.vimrc
```

У прикладі також припускається, що сценарій виконується користувачем root або користувачем `carol`, оскільки більшість файлів не зможуть прочитати інші користувачі. Якщо `script.sh` не знаходиться в каталозі, зазначеному в змінній середовища `PATH`, тоді його слід вказати з повним шляхом.

Вправи до посібника

1. Як можна використати команду `test`, щоб перевірити, чи шлях до файлу, що зберігається у змінній `FROM`, є новішим за шлях, який зберігається у змінній `TO`?

2. Наступний сценарій має виводити числову послідовність від 0 до 9, але замість цього він необмежено виводить 0. Що потрібно зробити, щоб отримати очікуваний результат?

```
#!/bin/bash

COUNTER=0

while [ $COUNTER -lt 10 ]
do
  echo $COUNTER
done
```

3. Припустимо, що користувач написав сценарій, який потрібен для сортування списку імен користувачів. Отриманий відсортований список представлений на його комп'ютері таким чином:

```
carol
Dave
emma
Frank
Grace
henry
```

Однак цей же список на комп'ютері його колеги сортується так:

```
Dave
Frank
Grace
carol
emma
henry
```

Чим можна пояснити відмінності між двома відсортованими списками?

Дослідницькі вправи

1. Як можна використати всі аргументи командного рядка сценарію для ініціалізації масиву Bash?

2. Чому інтуїтивно зрозуміло, що команда `test 1 > 2` оцінюється як true?

3. Як користувач тимчасово може змінити роздільник полів за замовчуванням лише на символ нового рядка, маючи при цьому можливість повернути його до вихідного значення?

Підсумки

У цьому уроці глибше розглядаються тести, доступні для команди `test`, а також інші умовні та циклічні конструкції, необхідні для написання більш складних сценаріїв оболонки. Простий сценарій синхронізації файлів наведено як приклад практичного застосування сценарію оболонки. Урок розглядає наступні питання

- Розширені тести для умовних конструкцій `if` і `case`.
- Циклічні конструкції оболонки: `for`, `until` і `while`.
- Ітерацію масивів і параметрів.

Розглянуті команди та процедури:

`test`

Виконує порівняння між елементами, наданими командою.

`if`

Логічна конструкція, яка використовується в сценаріях для оцінки чогось як істинного чи хибного, а потім виконання команди розгалуження на основі результатів.

`case`

оцінка кількох значень щодо однієї змінної. Виконання команди скрипта потім виконується в залежності від результату команди `case`.

`for`

повторне виконання команди на основі заданих критеріїв.

`until`

повторне виконання команди, доки вираз не отримає значення false.

`while`

повторне виконання команди, поки даний вираз оцінюється як істинний.

Відповіді до вправ посібника

1. Як можна використати команду `test`, щоб перевірити, чи шлях до файлу, що зберігається у змінній `FROM`, є новішим за шлях, який зберігається у змінній `TO`?

   Команда `test "$FROM" -nt "$TO"` поверне код статусу 0, якщо файл у змінній `FROM` є новішим за файл у змінній `TO`.

2. Наступний сценарій має виводити числову послідовність від 0 до 9, але замість цього він необмежено виводить 0. Що потрібно зробити, щоб отримати очікуваний результат?

```
#!/bin/bash

COUNTER=0

while [ $COUNTER -lt 10 ]
do
  echo $COUNTER
done
```

   Змінну `COUNTER` слід збільшити, що можна зробити за допомогою арифметичного виразу `COUNTER=$(( $COUNTER + 1 ))`, щоб зрештою досягти критеріїв зупинки та завершити цикл.

3. Припустимо, що користувач написав сценарій, якому потрібен відсортований список імен користувачів. Отриманий відсортований список представлений на його комп'ютері таким чином:

```
carol
Dave
emma
Frank
Grace
henry
```

   Однак цей же список на комп'ютері його колеги сортується так:

```
Dave
Frank
Grace
carol
```

```
emma
henry
```

Чим можна пояснити відмінності між двома відсортованими списками?

Сортування базується на поточній локалі системи. Щоб уникнути невідповідностей, завдання сортування слід виконувати зі змінною середовища `LANG`, встановленою на `C`.

Відповіді до дослідницьких вправ

1. Як можна використати всі аргументи командного рядка сценарію для ініціалізації масиву Bash?

   Команди `PARAMS=( $* )` або `PARAMS=( "$@" )` створять масив під назвою `PARAMS` з усіма аргументами.

2. Чому інтуїтивно зрозуміло, що команда `test 1 > 2` оцінюється як true?

   Оператор `>` призначений для використання з тестами рядків, без числових тестів.

3. Як користувач тимчасово може змінити роздільник полів за замовчуванням лише на символ нового рядка, маючи при цьому можливість повернути його до вихідного значення?

   Копію змінної `IFS` можна зберегти в іншій змінній: `OLDIFS=$IFS`. Тоді новий роздільник рядків визначається за допомогою `IFS=$'\n'`, а змінну IFS можна повернути назад за допомогою `IFS=$OLDIFS`.

Linux Professional Institute

Розділ 106: Інтерфейси користувача та робочі столи

106.1 Встановлення та налаштування X11

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 106.1

Ваговий коефіцієнт

2

Ключові галузі знань

- Розуміння архітектури X11.
- Базове розуміння та знання файлу конфігурації X Window.
- Зміна певних аспектів конфігурації Xorg, такі як розкладка клавіатури.
- Розуміння компонентів середовища робочого столу, таких як менеджери дисплеїв і менеджери вікон.
- Керування доступом до X-сервера та відображення програми на віддалених X-серверах.
- Розуміння Wayland.

Частковий список файлів, термінів та утиліт, що використовуються

- `/etc/X11/xorg.conf`
- `/etc/X11/xorg.conf.d/`
- `~/.xsession-errors`
- `xhost`
- `xauth`
- `DISPLAY`
- `X`

106.1 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	106 Інтерфейси користувача та робочі столи
Тема:	106.1 Встановлення та налаштування X11
Урок:	1 з 1

Вступ

X Window System — це програмний стек, який використовується для відображення тексту та графіки на екрані. Загальний вигляд і дизайн X-клієнта не визначаються системою X Window, а натомість обробляються кожним окремим X-клієнтом, *менеджером вікон* (наприклад, Window Maker, Tab Window Manager) або повним *середовищем робочого столу*, таким як KDE, GNOME або Xfce. Робочі середовища будуть розглянуті в наступному уроці. Цей урок буде зосереджений на основній архітектурі та загальних інструментах для системи X Window, які адміністратор використовуватиме для налаштування X.

Система X Window є кросплатформною та працює на різних операційних системах, таких як Linux, BSD, Solaris та інших Unix-подібних системах. Також доступні реалізації для macOS від Apple і Microsoft Windows.

Основною версією протоколу X, що використовується в сучасних дистрибутивах Linux, є *X.org* версія 11, зазвичай записується як *X11*. Протокол X — це механізм зв'язку між X-клієнтом і X-сервером. Відмінності між X-клієнтом і X-сервером будуть розглянуті нижче.

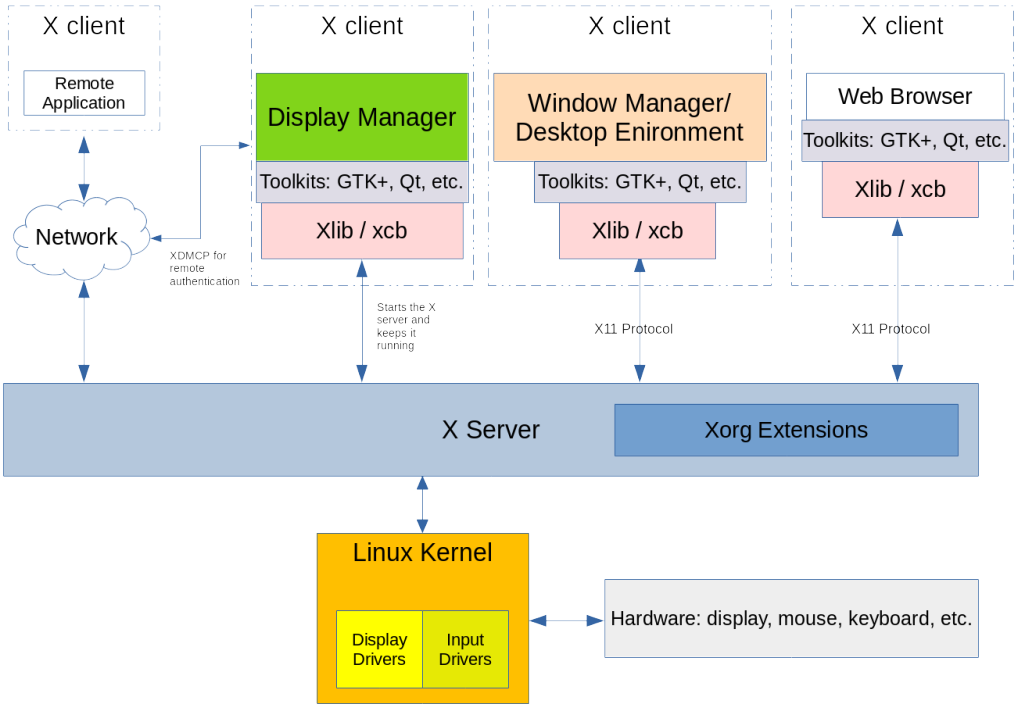
NOTE Попередником X Window System була віконна система під назвою *W*, яка була

спільною розробкою IBM, DEC і MIT. Це програмне забезпечення виникло в *Project Athena* в 1984 році. Коли розробники почали роботу над новим дисплеєм, вони вибрали наступну літеру англійського алфавіту: “X”. Еволюція системи X Window наразі контролюється *MIT X Consortium*.

Архітектура системи X Window

X Window System надає механізми для відтворення базових двовимірних форм (і тривимірних форм через розширення) на дисплеї. Вона розділена на клієнта та сервер, і в більшості установок, де потрібен графічний робочий стіл, обидва ці компоненти знаходяться на одному комп’ютері. Клієнтський компонент приймає форму програми, такої як емулятор терміналу, гра або веб-браузер. Кожна клієнтська програма інформує X-сервер про розташування та розмір свого вікна на екрані комп’ютера. Клієнт також обробляє те, що надходить у це вікно, а X-сервер розміщує запитуване зображення на екрані. Система X Window також обробляє вхідні дані з таких пристроїв, як миші, клавіатури, трекпади тощо.

Базова структура системи X Window



Система X Window підтримує роботу в мережі, і кілька X-клієнтів з різних комп’ютерів у мережі можуть надсилати запити на малювання до одного віддаленого X-сервера. Це

пояснюється тим, що адміністратор або користувач може отримати доступ до графічної програми на віддаленій системі, яка може бути недоступна в їхній локальній системі.

Ключовою особливістю системи X Window є її модульність. Протягом існування системи X Window було розроблено нові функції та додано до її структури. Ці нові компоненти були додані лише як розширення до X-сервера, залишивши основний протокол X11 без змін. Ці розширення містяться у файлах бібліотеки *Xorg*. Приклади бібліотек Xorg включають: `libXrandr`, `libXcursor`, `libX11`, `libxkbfile`, а також кілька інших, кожна з яких надає розширену функціональність для X-сервера.

Менеджер дисплея забезпечує графічний вхід до системи. Ця система може бути локальним комп'ютером або комп'ютером у мережі. Менеджер дисплея запускається після завантаження комп'ютера та запускає сеанс X-сервера для автентифікованого користувача. Менеджер дисплея також відповідає за підтримку та роботу X-сервера. Приклади менеджерів дисплеїв: GDM, SDDM і LightDM.

Кожен екземпляр запущеного X-сервера має *display name* для його ідентифікації. Display name містить наступне:

```
hostname:displaynumber.screennumber
```

Display name також вказує графічній програмі, де його слід відобразити та на якому хості (якщо використовується віддалене X-з'єднання).

`hostname` відноситься до назви системи, яка відображатиме програму. Якщо hostname відсутнє у display name, тоді передбачається localhost.

`displaynumber` посилається на колекцію "екранів", які використовуються, чи то один екран ноутбука, чи кілька екранів на робочій станції. Кожному запущеному сеансу X-сервера надається display number, починаючи з `0`.

`screennumber` за умовчанням дорівнює `0`. Це може бути, якщо є лише один фізичний екран або кілька фізичних екранів налаштовані для роботи як один екран. Коли всі екрани в налаштуваннях із кількома моніторами об'єднані в один логічний екран, вікна програм можна вільно переміщувати між екранами. У ситуаціях, коли кожен екран налаштовано на роботу незалежно один від одного, на кожному екрані будуть розміщені вікна програм, які відкриваються в них, і вікна не можна переміщати з одного екрана на інший. Кожен дисплейний екран матиме власний номер. Якщо використовується лише один логічний екран, то крапка та номер екрана опускаються.

Display name запущеного X-сеансу зберігається в змінній середовища `DISPLAY`:

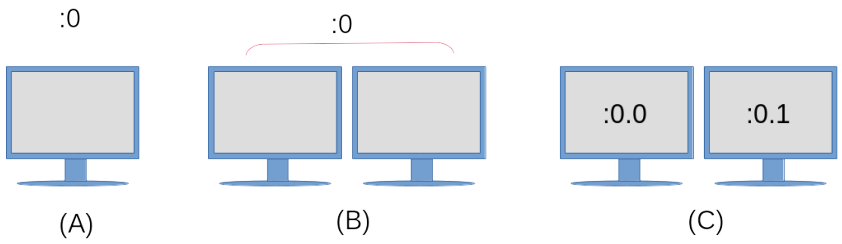
```
$ echo $DISPLAY
:0
```

Наведемо детальний опис отриманого результату:

1. X-сервер, який використовується, знаходиться в локальній системі, тому ліворуч від двокрапки нічого не друкується.
2. Поточний сеанс X-сервера є першим, на що вказує `0` відразу після двокрапки.
3. Використовується лише один логічний екран, тому номер екрана не видно.

Щоб детальніше проілюструвати цю концепцію, зверніться до наступної діаграми:

Приклад конфігурації дисплея



(A): Один монітор із конфігурацією одного дисплея та лише одним екраном. (B): Налаштовано як один дисплей із двома фізичними моніторами, налаштованими як один екран. Вікна програм можна вільно переміщати між двома моніторами. (C):: Конфігурація єдиного дисплея (на що вказує `:0`), однак кожен монітор є незалежним екраном. Обидва екрани використовуватимуть ті самі пристрої введення, наприклад, клавіатуру та мишу, однак програму, відкриту на екрані `:0.0`, не можна перемістити на екран `:0.1` і навпаки.

Щоб запустити програму на певному екрані, призначте номер екрана змінній середовища `DISPLAY` перед запуском програми:

```
$ DISPLAY=:0.1 firefox &
```

Ця команда запустить веб-браузер Firefox на екрані праворуч на діаграмі вище. Деякі набори інструментів також надають параметри командного рядка, щоб вказати програмі працювати на певному екрані. Перегляньте `--screen` і `--display` на man-сторінці `gtk-options(7)` для прикладу.

Конфігурація X-сервера

Традиційно основним конфігураційним файлом, який використовується для налаштування X-сервера, є файл `/etc/X11/xorg.conf`. У сучасних дистрибутивах Linux X-сервер налаштовуватиме себе під час свого запуску, тому файл `xorg.conf` може не існувати.

Файл `xorg.conf` розділений на окремі частини, які називаються *розділи* (sections). Кожен розділ починається з терміну `Section`, а після цього терміну йде *ім'я розділу*, яке вказує на конфігурацію компонента. Кожен `Section` завершується відповідним `EndSection`. Типовий файл `xorg.conf` містить такі розділи:

`InputDevice`

Використовується для налаштування певної моделі клавіатури чи миші.

`InputClass`

У сучасних дистрибутивах Linux цей розділ зазвичай знаходиться в окремому файлі конфігурації, розташованому в папці `/etc/X11/xorg.conf.d/`. `InputClass` використовується для налаштування *класу* апаратних пристроїв, таких як клавіатури та миші, а не окремого апаратного забезпечення. Нижче наведено приклад файлу `/etc/X11/xorg.conf.d/00-keyboard.conf`:

```
Section "InputClass"
        Identifier "system-keyboard"
        MatchIsKeyboard "on"
        Option "XkbLayout" "us"
        Option "XkbModel" "pc105"
EndSection
```

Параметр для `XkbLayout` визначає розкладку клавіш на клавіатурі, наприклад Дворака, для лівої чи правої руки, QWERTY і мову. Параметр для `XkbModel` використовується для визначення типу клавіатури, що використовується. Таблицю моделей, розкладок та їхніх описів можна знайти в `xkeyboard-config(7)`. Файли, пов'язані з розкладками клавіатури, можна знайти в `/usr/share/X11/xkb`. Приклад грецької політонічної розкладки клавіатури на комп'ютері Chromebook буде мати такий вигляд:

```
Section "InputClass"
        Identifier "system-keyboard"
        MatchIsKeyboard "on"
        Option "XkbLayout" "gr(polytonic)"
```

```
        Option "XkbModel" "chromebook"
EndSection
```

Крім того, розкладку клавіатури можна змінити під час запущеного сеансу X за допомогою команди `setxkbmap`. Ось приклад цієї команди, яка налаштовує грецький політонічний макет на комп'ютері Chromebook:

```
$ setxkbmap -model chromebook -layout "gr(polytonic)"
```

Це налаштування діятиме лише доки використовується сеанс X. Щоб зробити такі зміни постійними, змініть файл `/etc/X11/xorg.conf.d/00-keyboard.conf`, включивши необхідні налаштування.

NOTE Команда `setxkbmap` використовує *X Keyboard Extension* (XKB). Це приклад додаткової функціональності системи X Window завдяки використанню розширень.

Сучасні дистрибутиви Linux надають команду `localectl` через `systemd`, яка ітакож може використовувати для зміни розкладки клавіатури та автоматично створює файл конфігурації `/etc/X11/xorg.conf.d/00-keyboard.conf`. Ось ще раз приклад налаштування грецької політонічної клавіатури на Chromebook, цього разу за допомогою команди `localectl`:

```
$ localectl --no-convert set-x11-keymap "gr(polytonic)" chromebook
```

Параметр `--no-convert` використовується тут, щоб завадити `localectl` змінювати розкладку клавіш консолі хоста.

`Monitor`

Розділ `Monitor` описує фізичний монітор, який використовується, і місце, куди його підключено. Ось приклад конфігурації апаратного монітора, підключеного до другого порту дисплея, який використовується як основний монітор.

```
Section "Monitor"
        Identifier  "DP2"
        Option      "Primary" "true"
EndSection
```

`Device`

У розділі `Device` описується фізична відеокарта, яка використовується. Розділ також містить модуль ядра, який використовується як драйвер для відеокарти, а також його фізичне розташування на материнській платі.

```
Section "Device"
        Identifier  "Device0"
        Driver      "i915"
        BusID       "PCI:0:2:0"
EndSection
```

`Screen`

Розділ `Screen` об'єднує розділи `Monitor` і `Device`. Приклад розділу `Screen` може мати такий вигляд:

```
Section "Screen"
        Identifier "Screen0"
        Device     "Device0"
        Monitor    "DP2"
EndSection
```

`ServerLayout`

Розділ `ServerLayout` об'єднує всі розділи, такі як миша, клавіатура та екрани, в один інтерфейс X Window System.

```
Section "ServerLayout"
    Identifier   "Layout-1"
    Screen       "Screen0" 0 0
    InputDevice  "mouse1"  "CorePointer"
    InputDevice  "system-keyboard"  "CoreKeyboard"
EndSection
```

NOTE У конфігураційному файлі можна знайти не всі розділи. У випадках, коли розділ відсутній, значення за замовчуванням надаються запущеним екземпляром X-сервера.

Зазначені користувачем конфігураційні файли також знаходяться в `/etc/X11/xorg.conf.d/`. Файли конфігурації, які надаються дистрибутивом, знаходяться в `/usr/share/X11/xorg.conf.d/`. Конфігураційні файли, розташовані в

`/etc/X11/xorg.conf.d/`, аналізуються перед файлом `/etc/X11/xorg.conf`, якщо він існує в системі.

Команда `xdpyinfo` використовується на комп'ютері для відображення інформації про запущений екземпляр X-сервера. Нижче наведено зразок результату команди:

```
$ xdpyinfo
name of display:    :0
version number:    11.0
vendor string:    The X.Org Foundation
vendor release number:    12004000
X.Org version: 1.20.4
maximum request size:  16777212 bytes
motion buffer size:  256
bitmap unit, bit order, padding:    32, LSBFirst, 32
image byte order:    LSBFirst
number of supported pixmap formats:    7
supported pixmap formats:
    depth 1, bits_per_pixel 1, scanline_pad 32
    depth 4, bits_per_pixel 8, scanline_pad 32
    depth 8, bits_per_pixel 8, scanline_pad 32
    depth 15, bits_per_pixel 16, scanline_pad 32
    depth 16, bits_per_pixel 16, scanline_pad 32
    depth 24, bits_per_pixel 32, scanline_pad 32
    depth 32, bits_per_pixel 32, scanline_pad 32
keycode range:    minimum 8, maximum 255
focus:  None
number of extensions:    25
    BIG-REQUESTS
    Composite
    DAMAGE
    DOUBLE-BUFFER
    DRI3
    GLX
    Generic Event Extension
    MIT-SCREEN-SAVER
    MIT-SHM
    Present
    RANDR
    RECORD
    RENDER
    SECURITY
    SHAPE
    SYNC
```

```
    X-Resource
    XC-MISC
    XFIXES
    XFree86-VidModeExtension
    XINERAMA
    XInputExtension
    XKEYBOARD
    XTEST
    XVideo
default screen number:    0
number of screens:    1

screen #0:
  dimensions:    3840x1080 pixels (1016x286 millimeters)
  resolution:    96x96 dots per inch
  depths (7):    24, 1, 4, 8, 15, 16, 32
  root window id:    0x39e
  depth of root window:    24 planes
  number of colormaps:    minimum 1, maximum 1
  default colormap:    0x25
  default number of colormap cells:    256
  preallocated pixels:    black 0, white 16777215
  options:    backing-store WHEN MAPPED, save-unders NO
  largest cursor:    3840x1080
  current input event mask:    0xda0033
    KeyPressMask            KeyReleaseMask          EnterWindowMask
    LeaveWindowMask         StructureNotifyMask     SubstructureNotifyMask
    SubstructureRedirectMask PropertyChangeMask      ColormapChangeMask
  number of visuals:    270
...
```

Більш релевантні частини результату виділено жирним шрифтом, наприклад, display name (таке саме, як і вміст змінної середовища `DISPLAY`), інформація про версію X-сервера, що використовується, кількість і список Xorg-розширень, які використовуються, та додаткова інформація про сам екран.

Створення базового файлу Xorg-конфігурації

Незважаючи на те, що X створить свою конфігурацію після запуску системи на встановлених сучасних Linux, файл `xorg.conf` все ще можна використовувати. Щоб створити постійний файл `/etc/X11/xorg.conf`, виконайте таку команду:

```
$ sudo Xorg -configure
```

NOTE

Якщо сеанс X вже запущено, вам потрібно буде вказати інший `DISPLAY` у вашій команді, наприклад:

```
$ sudo Xorg :1 -configure
```

У деяких дистрибутивах Linux команду `X` можна використовувати замість `Xorg`, оскільки `X` є символьним посиланням на `Xorg`.

У вашому поточному робочому каталозі буде створено файл xorg.conf.new. Вміст цього файлу отримано з того, що X-сервер виявив доступним в апаратному забезпеченні та драйверах локальної системи. Щоб використовувати цей файл, його потрібно буде перемістити до каталогу `/etc/X11/` і перейменувати на `xorg.conf`:

```
$ sudo mv xorg.conf.new /etc/X11/xorg.conf
```

NOTE

Наступні man-сторінки містять додаткову інформацію про компоненти системи X Window: `xorg.conf(5)`, `Xserver(1)`, `X(1)` та `Xorg(1)`.

Wayland

Wayland — це новий протокол відображення, призначений для заміни системи X Window. Багато сучасних дистрибутивів Linux використовують його як сервер відображення за умовчанням. Він потребує менше системних ресурсів та займає менше місця при встановленні, ніж X. Проект розпочався у 2010 році та все ще перебуває в стадії активного розвитку, включаючи роботу активних і колишніх розробників X.org.

На відміну від X Window System, немає екземпляра сервера, який працює між клієнтом і ядром. Натомість вікно клієнта працює з власним кодом або кодом інструментарію (наприклад, Gtk+ або Qt), щоб забезпечити рендеринг. Щоб виконати візуалізацію, до ядра Linux надсилається запит через протокол Wayland. Ядро пересилає запит через протокол Wayland до Wayland *композитору*, який обробляє введення з пристрою, керування вікнами та композицію. Композитор — це частина системи, яка поєднує відтворені елементи у візуальний вихід на екрані.

Більшість сучасних наборів інструментів, таких як Gtk+ 3 і Qt 5, було оновлено, щоб дозволити рендеринг або в системі X Window, або на комп'ютері, на якому запущено Wayland. Наразі не всі автономні програми були написані для підтримки візуалізації у

Wayland. Для програм і фреймворків, які все ще орієнтовані на роботу системи X Window, програма може працювати в *XWayland*. Система XWayland — це окремий X-сервер, який працює всередині клієнта Wayland і, таким чином, відображає вміст вікна клієнта в окремому екземплярі X-сервера.

Подібно до того, як система X Window використовує змінну середовища `DISPLAY` для відстеження екранів, що використовуються, протокол Wayland використовує змінну середовища `WAYLAND_DISPLAY`. Нижче наведено зразок вихідних даних із системи, на якій працює дисплей Wayland:

```
$ echo $WAYLAND_DISPLAY
wayland-0
```

Ця змінна середовища недоступна в системах під керуванням X.

Вправи до посібника

1. Яку команду ви використаєте, щоб визначити, які розширення Xorg доступні в системі?

2. Ви щойно отримали абсолютно нову 10-кнопкову мишу для свого комп'ютера, однак її потрібно буде додатково налаштувати, щоб усі кнопки працювали належним чином. Не змінюючи решту конфігурації X-сервера, який каталог ви б використали для створення нового файлу конфігурації для цієї миші, і який конкретний розділ конфігурації використовувався б у цьому файлі?

3. Який компонент встановленого Linux відповідає за підтримку роботи X-сервера?

4. Який параметр командного рядка використовується з командою X для створення нового файлу конфігурації `xorg.conf`?

Дослідницькі вправи

1. Яким буде вміст змінної середовища `DISPLAY` у системі з назвою `lab01`, яка використовує одну конфігурацію дисплея? Припустімо, що змінна середовища `DISPLAY` переглядається в емуляторі терміналу на третьому дисплейному екрані.

2. Яку команду можна використати для створення файлу конфігурації клавіатури для використання системою X Window?

3. При стандартному встановленні Linux користувач може переключитися на віртуальний термінал, натиснувши клавіші `Ctrl` + `Alt` + `F1`-`F6` на клавіатурі. Вас попросили налаштувати систему кіоску з графічним інтерфейсом і цю функцію потрібно вимкнути, щоб запобігти несанкціонованому втручанню в систему. Ви вирішили створити файл конфігурації `/etc/X11/xorg.conf.d/10-kiosk.conf`. Використовуючи розділ `ServerFlags` (який використовується для встановлення глобальних параметрів Xorg на сервері), який параметр потрібно вказати? Перегляньте man-сторінку `xorg(1)`, щоб знайти потрібну опцію.

Підсумки

У цьому уроці розглядалася система X Window, яка використовується в Linux. X Window System використовується для відтворення зображень і тексту на екранах, як вони визначені в різних конфігураційних файлах. Система X Window часто використовується для налаштування пристроїв введення, таких як миші та клавіатури. На цьому уроці обговорювалися такі моменти:

- Архітектура X Window System на високому рівні.
- Які файли конфігурації використовуються для налаштування системи X Windows і їх розташування у файловій системі.
- Як використовувати змінну середовища `DISPLAY` у системі з X.
- Короткий вступ до дисплейного протоколу Wayland.

Були розглянуті наступні команди та конфігураційні файли:

- Зміна розкладки клавіатури в інсталяції Xorg за допомогою `setxkbmap` і `localectl`.
- Команда `Xorg` для створення нового файлу конфігурації `/etc/X11/xorg.conf`.
- Вміст конфігураційних файлів Xorg, розташованих: `/etc/X11/xorg.conf`, `/etc/X11/xorg.conf.d/` та `/usr/share/X11/xorg.conf.d/`.
- Команда `xdpyinfo` для відображення загальної інформації про запущений сеанс X-сервера.

Відповіді до вправ посібника

1. Яку команду ви використаєте, щоб визначити, які розширення Xorg доступні в системі?

   ```
   $ xdpyinfo
   ```

2. Ви щойно отримали абсолютно нову 10-кнопкову мишу для свого комп'ютера, однак її потрібно буде додатково налаштувати, щоб усі кнопки працювали належним чином. Не змінюючи решту конфігурації X-сервера, який каталог ви б використали для створення нового файлу конфігурації для цієї миші, і який конкретний розділ конфігурації використовувався б у цьому файлі?

   Визначені користувачем конфігурації мають бути розташовані в `/etc/X11/xorg.conf.d/`, а окремим розділом, необхідним для цієї конфігурації миші, буде `InputDevice`.

3. Який компонент встановленого Linux відповідає за підтримку роботи X-сервера?

   Менеджер дисплея.

4. Який параметр командного рядка використовується з командою `X` для створення нового файлу конфігурації `xorg.conf`?

   ```
   -configure
   ```

   Пам'ятайте, що команда `X` є символьним посиланням на команду `Xorg`.

Відповіді до дослідницьких вправ

1. Яким буде вміст змінної середовища `DISPLAY` у системі з назвою `lab01`, яка використовує одну конфігурацію дисплея? Припустімо, що змінна середовища `DISPLAY` переглядається в емуляторі терміналу на третьому дисплейному екрані.

```
$ echo $DISPLAY
lab01:0.2
```

2. Яку команду можна використати для створення файлу конфігурації клавіатури для використання системою X Window?

```
$ localectl
```

3. При стандартному встановленні Linux користувач може переключитися на віртуальний термінал, натиснувши клавіші `Ctrl` + `Alt` + `F1`-`F6` на клавіатурі. Вас попросили налаштувати систему кіоску з графічним інтерфейсом і цю функцію потрібно вимкнути, щоб запобігти несанкціонованому втручанню в систему. Ви вирішили створити файл конфігурації `/etc/X11/xorg.conf.d/10-kiosk.conf`. Використовуючи розділ `ServerFlags` (який використовується для встановлення глобальних параметрів Xorg на сервері), який параметр потрібно вказати? Перегляньте man-сторінку `xorg(1)`, щоб знайти опцію.

```
Section "ServerFlags"
    Option  "DontVTSwitch"  "True"
EndSection
```

106.2 Графічні робочі столи

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 106.2

Ваговий коефіцієнт

1

Ключові галузі знань

- Уявлення про основні робочі середовища.
- Знання протоколів доступу до сеансів віддаленого робочого столу.

Частковий список файлів, термінів та утиліт, що використовуються

- KDE
- Gnome
- Xfce
- X11
- XDMCP
- VNC
- Spice
- RDP

106.2 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	106 Інтерфейси користувача та робочі столи
Тема:	106.2 Графічні робочі столи
Урок:	1 з 1

Вступ

Операційні системи на основі Linux відомі своїм вдосконаленим інтерфейсом командного рядка, але це може лякати користувачів, які не мають технічних знань. З метою зробити користування комп'ютером більш інтуїтивно зрозумілим, поєднання дисплеїв високої роздільної здатності з вказівними пристроями призвело до появи інтерфейса користувача, що керується зображеннями. У той час як для інтерфейсу командного рядка потрібні попередні знання про назви програм та параметри їх конфігурації, за допомогою *графічного інтерфейсу користувача* (GUI) програмні функції можна запустити, вказуючи на знайомі візуальні елементи, що робить криву навчання менш крутою. Крім того, графічний інтерфейс найкраще підходить для мультимедіа та інших візуально орієнтованих видів діяльності.

Дійсно, графічний інтерфейс користувача є майже синонімом комп'ютерного інтерфейсу, і більшість дистрибутивів Linux постачаються з графічним інтерфейсом, встановленим за замовчуванням. Однак не існує єдиної монолітної програми, відповідальної за повнофункціональні графічні робочі столи, доступні в системах Linux. Натомість кожен графічний стіл насправді є великою колекцією програм та їхніх залежностей, які варіюються від вибору дистрибутива до особистого смаку користувача.

Система X Window

У Linux та інших Unix-подібних операційних системах, де вона використовується, *Система X Window* (також відома як *X11* або просто *X*) надає ресурси низького рівня, пов'язані з відтворенням графічного інтерфейсу та взаємодією користувача з ним, наприклад:

- Обробка подій введення, таких як рухи миші або натискання клавіш.
- Можливість вирізати, копіювати та вставляти текстовий вміст між окремими програмами.
- Програмний інтерфейс інших програм використовується для малювання графічних елементів.

Хоча система X Window відповідає за керування графічним дисплеєм (сам відеодрайвер є частиною X), вона не призначена для малювання складних візуальних елементів самостійно. Фігури, кольори, відтінки та будь-які інші візуальні ефекти генеруються програмою, що працює поверх X. Такий підхід дає програмам багато простору для створення інтерфейсів, які можна налаштовувати, але також може призвести до накладних витрат на розробку, які виходять за межі програми та до невідповідності зовнішнього вигляду та поведінки в порівнянні з іншими програмними інтерфейсами.

З точки зору розробника, запровадження *середовища робочого столу* полегшує програмування графічного інтерфейсу користувача, пов'язане з базовою розробкою програми, тоді як з точки зору користувача це дає узгоджену роботу з різними програмами. Настільні середовища об'єднують інтерфейси програмування, бібліотеки та допоміжні програми, які співпрацюють для реалізації традиційних середовищ, які ще досі постійно розвиваються.

Середовище робочого столу

Традиційний GUI настільного комп'ютера складається з різних вікон. Термін *вікно* тут використовується для позначення будь-якої автономної області екрана, пов'язаної з запущеними процесами. Оскільки сама система X Window пропонує лише базові інтерактивні функції, повна взаємодія з користувачем залежить від компонентів середовища робочого столу.

Ймовірно, найважливіший компонент середовища робочого столу, *менеджер вікон* контролює розміщення вікон і їх оформлення. Саме менеджер вікон додає рядок заголовка до вікна, кнопки керування, зазвичай пов'язані з діями мінімізації, розгортання та закриття, і керує перемиканням між відкритими вікнами.

NOTE

Основні концепції, які можна знайти в графічних інтерфейсах настільних комп'ютерів, виникли з ідей, узятих із реальних офісних робочих просторів. Образно кажучи, екран комп'ютера – це робочий стіл, на якому розміщені такі об'єкти, як документи та папки. Вікно програми з вмістом документа імітує фізичні дії, такі як заповнення форми або малювання малюнка. Як і фактичні робочі столи, комп'ютерні настільні комп'ютери також мають програмні аксесуари, такі як блокноти, годинники, календарі тощо, більшість із яких засновані на їхніх «реальних» аналогах.

Усі робочі середовища надають менеджер вікон, який відповідає зовнішньому вигляду його *набору інструментів для віджетів*. Віджети — це інформаційні або інтерактивні візуальні елементи, такі як кнопки або поля введення тексту, розташовані у вікні програми. Стандартні компоненти робочого столу, наприклад, засіб запуску додатків, панель завдань тощо, і сам менеджер вікон покладаються на такі набори інструментів для створення своїх інтерфейсів.

Бібліотеки програмного забезпечення, такі як *GTK+* і *Qt*, надають віджети, які програмісти можуть використовувати для створення складних графічних інтерфейсів для своїх програм. Історично склалося так, що програми, розроблені за допомогою GTK+, не виглядали як програми, створені за допомогою Qt, і навпаки, але краща підтримка тем у сучасних настільних середовищах робить цю різницю менш очевидною.

Загалом GTK+ і Qt пропонують однакові функції щодо віджетів. Прості інтерактивні елементи можуть бути схожими, тоді як складені віджети, такі як діалогове вікно, яке використовується програмами для відкриття або збереження файлів, можуть виглядати зовсім інакше. Тим не менш, програми, створені за допомогою різних наборів інструментів, можуть працювати одночасно, незалежно від того, який набір інструментів віджетів використовують інші компоненти робочого столу.

На додаток до базових компонентів робочого столу, які можна вважати окремими самостійними програмами, середовище робочого столу реалізує метафору робочого столу, надаючи мінімальний набір додаткових програм, розроблених відповідно до тих самих інструкцій щодо дизайну. Варіанти наведених нижче програм зазвичай надаються в усіх основних настільних середовищах:

Системні програми

Емулятор терміналу, файловий менеджер, менеджер встановлення пакунків, засоби налаштування системи.

Зв'язок та Інтернет

Менеджер контактів, поштовий клієнт, веб-браузер.

Офісні програми

Календар, калькулятор, текстовий редактор.

Середовища настільного комп'ютера можуть включати багато інших служб і програм: засіб привітання екрана входу, менеджер сеансів, міжпроцесний зв'язок, агент зв'язку ключів тощо. Вони також містять функції, надані сторонніми системними службами, такими як *PulseAudio* для звуку та *CUPS* для друку. Для роботи цих функцій не потрібне графічне середовище, але середовище робочого столу надає графічні інтерфейси для полегшення налаштування та роботи таких ресурсів.

Популярні робочі середовища

Багато пропрієтарних операційних систем підтримують лише одне офіційне середовище робочого столу, яке прив'язане до їх конкретного випуску та не має змінюватися. На відміну від них, операційні системи на базі Linux підтримують різні варіанти середовища робочого столу, які можна використовувати разом із X. Кожне середовище робочого столу має свої особливості, але вони зазвичай мають спільні концепції дизайну:

- Засіб запуску програм із переліком вбудованих і сторонніх програм, доступних у системі.
- Правила, що визначають програми за замовчуванням, пов'язані з типами файлів і протоколами.
- Інструменти конфігурації для налаштування зовнішнього вигляду та поведінки робочого середовища.

Gnome — одне з найпопулярніших робочих середовищ, яке є першим вибором у таких дистрибутивах, як Fedora, Debian, Ubuntu, SUSE Linux Enterprise, Red Hat Enterprise Linux, CentOS тощо. У своїй версії 3 Gnome значно змінив свій вигляд і структуру , відходячи від метафори робочого столу та представляючи *Gnome Shell* як новий інтерфейс.

Figure 1. Gnome Shell Activities

Загальний повноекранний засіб запуску *Gnome Shell Activities* замінив традиційний засіб запуску програм і панель завдань. Проте все ще можна використовувати Gnome 3 зі старим виглядом, вибравши опцію *Gnome Classic* на екрані входу.

KDE — це велика екосистема програм і платформа розробки. Його остання версія робочого середовища, *KDE Plasma*, використовується за замовчуванням у openSUSE, Mageia, Kubuntu тощо. Використання бібліотеки Qt є вражаючою особливістю KDE, що надає йому бездоганного вигляду та безліч оригінальних програм. KDE навіть надає інструмент налаштування для забезпечення візуального зв'язку з програмами GTK+.

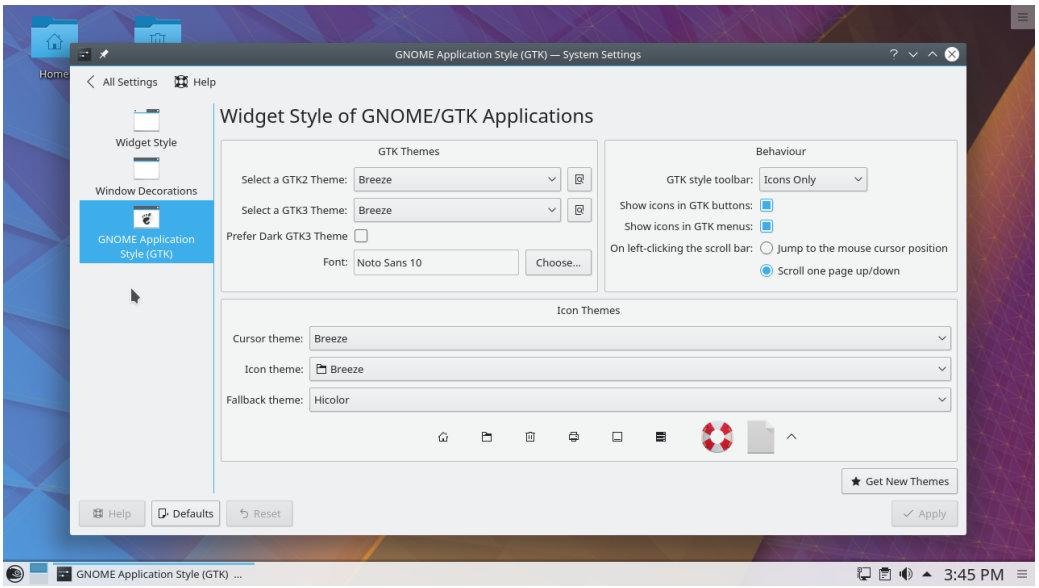


Figure 2. Налаштування KDE GTK

Xfce — це середовище робочого столу, яке має бути естетично привабливим, але не споживає багато ресурсів комп'ютера. Його структура дуже модульна, що дозволяє користувачеві активувати та деактивувати компоненти відповідно до потреб і вподобань користувача.

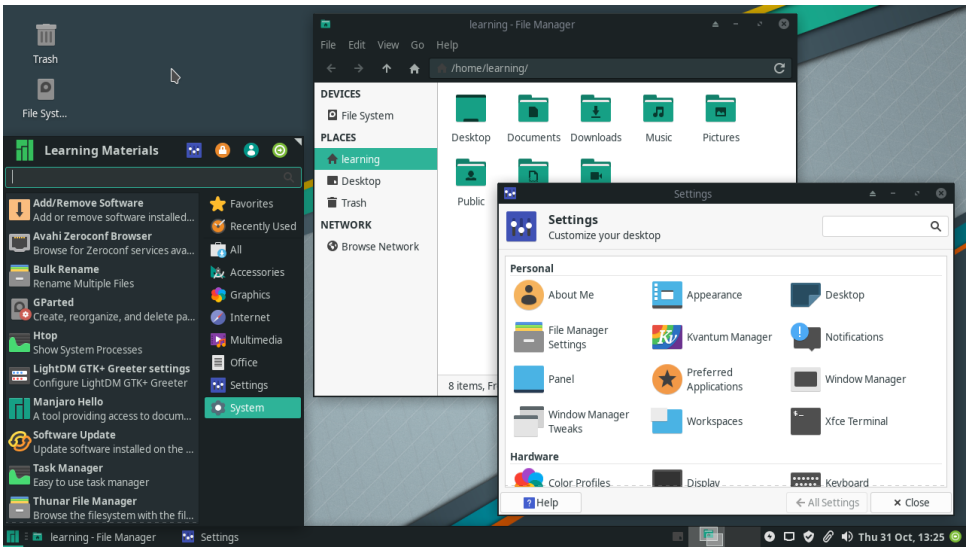


Figure 3. Робочий стіл Xfce

Існує багато інших робочих середовищ для Linux, які зазвичай надаються альтернативними розповсюджувачами. Дистрибутив Linux Mint, наприклад, надає два оригінальних робочих середовища: *Cinnamon* (форк Gnome 3) і *MATE* (форк Gnome 2). *LXDE* — це робоче середовище, адаптоване до низького споживання ресурсів, що робить його хорошим вибором для встановлення на старішому обладнанні або одноплатних комп'ютерах. Незважаючи на те, що LXDE не пропонує всіх функцій важких робочих середовищ, LXDE пропонує всі основні функції, які очікуються від сучасного графічного інтерфейсу користувача.

TIP

Комбінації клавіш відіграють важливу роль у взаємодії з робочим середовищем. Деякі комбінації клавіш, наприклад `Alt` + `Tab` для перемикання між вікнами або `Ctrl` + `c` для копіювання тексту, є універсальними для робочого середовища, але кожне робоче середовище також має власні комбінації клавіш. Комбінації клавіш можна знайти в інструменті конфігурації клавіатури, що надається середовищем робочого столу, де комбінації клавіш можна додавати або змінювати.

Взаємодія робочого столу

Різноманітність настільних середовищ в операційних системах на базі Linux накладає виклик: як змусити їх правильно працювати з графічними програмами або системними

службами сторонніх розробників без необхідності впровадження спеціальної підтримки для кожного з них. Спільні методи та специфікації в середовищах робочого столу значно покращують взаємодію з користувачем і вирішують багато питань розробки, оскільки графічні програми повинні взаємодіяти з поточним середовищем робочого столу незалежно від того, для якого середовища робочого столу вони спочатку були розроблені. Крім того, важливо зберегти загальні параметри робочого столу, якщо користувач зрештою змінить вибір середовища робочого столу.

Організація *freedesktop.org* підтримує велику кількість специфікацій для взаємодії робочих столів. Прийняття повної специфікації не є обов'язковим, але багато з них широко використовуються:

Адреси каталогів

Місце розміщення особистих налаштувань та інші файлів користувача.

Елементи робочого столу

Програми командного рядка можна запускати в середовищі робочого столу через будь-який емулятор терміналу, але було б надто заплутано зробити всі їх доступними в панелі запуску програм. Елементи робочого столу – це текстові файли, що закінчуються на `.desktop`, які використовуються середовищем робочого столу для збору інформації про доступні програми робочого столу та способи їх використання.

Автозапуск програм

Записи на робочому столі, що вказують програму, яка має запускатися автоматично після входу користувача.

Перетягування (Drag and drop)

Як програми повинні обробляти події перетягування.

Кошик

Загальне розташування файлів, видалених файловим менеджером, а також способи зберігання та видалення файлів звідти.

Теми значків

Загальний формат для взаємозамінних бібліотек значків.

Простота використання, яку забезпечують настільні середовища, має недолік порівняно з текстовими інтерфейсами, такими як оболонка: можливість надавати віддалений доступ. Хоча до середовища командного рядка віддаленої машини можна легко отримати доступ за допомогою таких інструментів, як `ssh`, віддалений доступ до графічних середовищ потребує інших методів і може не досягти задовільної продуктивності на повільніших

з'єднаннях.

Нелокальний доступ

Система X Window використовує дизайн, заснований на автономних *дисплеях*, де один і той самий *X дисплей-менеджер* може керувати кількома сеансами графічного робочого столу одночасно. По суті, дисплей аналогічний текстовому терміналу: обидва стосуються машини або програмного забезпечення, яке використовується як точка входу для встановлення незалежного сеансу операційної системи. Хоча найпоширеніші налаштування включають окремий графічний сеанс, запущений на локальній машині, також можливі інші менш звичайні налаштування:

- Перемикання між активними сеансами графічного робочого столу на одній машині.
- Більш ніж один набір пристроїв відображення (наприклад, екран, клавіатура, миша), підключених до однієї машини, кожен з яких керує власним сеансом графічного робочого столу.
- Сеанси віддаленого графічного робочого столу, коли графічний інтерфейс надсилається через мережу на віддалений дисплей.

Сеанси віддаленого робочого столу підтримуються X, який використовує *X Display Manager Control Protocol* (XDMCP) для зв'язку з віддаленими дисплеями. Через високе використання пропускної здатності XDMCP рідко використовується через Інтернет або в низькошвидкісних локальних мережах. Проблеми безпеки також викликають занепокоєння з XDMCP: локальний дисплей спілкується з привілейованим віддаленим дисплейним менеджером X для виконання віддалених процедур, тому потенційна вразливість може зробити можливим виконання довільних привілейованих команд на віддаленій машині.

Крім того, для XDMCP потрібні екземпляри X, запущені на обох кінцях з'єднання, що може зробити його неможливим, якщо система X Windows недоступна для всіх залучених машин. На практиці для встановлення сеансів віддаленого графічного робочого столу використовуються інші більш ефективні та менш інвазивні методи.

Virtual Network Computing (VNC) — це незалежний від платформи інструмент для перегляду та керування середовищами віддаленого робочого столу за допомогою протоколу *Remote Frame Buffer* (RFB). Через нього події, створені локальною клавіатурою та мишею, передаються на віддалений робочий стіл, який, у свою чергу, надсилає назад будь-які оновлення екрана для локального відображення. На одній машині можна запускати багато серверів VNC, але кожному серверу VNC потрібен ексклюзивний TCP-порт у мережевому інтерфейсі, який приймає вхідні запити сеансу. Згідно з угодою, перший

сервер VNC повинен використовувати TCP-порт 5900, другий — 5901 і так далі.

Сервер VNC не потребує спеціальних привілеїв для роботи. Звичайний користувач може, наприклад, увійти у свій віддалений обліковий запис і запустити звідти власний сервер VNC. Тоді на локальній машині будь-який клієнтський додаток VNC можна використовувати для доступу до віддаленого робочого столу (припускаючи, що відповідні мережеві порти доступні). Файл `~/.vnc/xstartup` — це сценарій оболонки, який виконується сервером VNC під час його запуску та може використовуватися для визначення середовища робочого столу, яке сервер VNC зробить доступним для клієнта VNC. Важливо зазначити, що VNC не надає сучасні методи шифрування та автентифікації, тому його слід використовувати разом із програмою стороннього розробника, яка надає такі функції. Для захисту з'єднань VNC часто використовуються методи, що включають тунелі VPN і SSH.

Remote Desktop Protocol (RDP) в основному використовується для віддаленого доступу до робочого столу операційної системи *Microsoft Windows* через мережевий порт TCP 3389. Незважаючи на те, що він використовує власний протокол RDP Microsoft, клієнтська реалізація, що використовується в системах Linux, є програмами з відкритим кодом, ліцензованими згідно з *GNU General Public License* (GPL) які не мають юридичних обмежень щодо використання.

Simple Protocol for Independent Computing Environments (Spice) містить набір інструментів, спрямованих на доступ до робочого середовища *віртуалізованих* систем на локальній машині або у віддаленому місці. Крім того, протокол Spice пропонує власні функції для інтеграції локальної та віддаленої систем, таких як можливість доступу до локальних пристроїв (наприклад, звукових динаміків і підключених пристроїв USB) із віддаленої машини та обмін файлами між двома системами.

Існують спеціальні клієнтські команди для підключення до кожного з цих протоколів віддаленого робочого столу, але клієнт віддаленого робочого столу *Remmina* надає вбудований графічний інтерфейс, який полегшує процес підключення, за бажанням зберігаючи налаштування підключення для подальшого використання. У Remmina є плагіни для кожного окремого протоколу, а також плагіни для XDMCP, VNC, RDP і Spice. Вибір правильного інструменту залежить від операційних систем, які використовуються, якості мережевого з'єднання та того, які функції середовища віддаленого робочого столу повинні бути доступні.

Вправи до посібника

1. Який тип програми забезпечує сеанси віконної оболонки в робочому середовищі?

2. Через різноманітність настільних середовищ Linux одна і та сама програма може мати більше однієї версії, кожна з яких найкраще підходить для певного інструментарію віджетів. Наприклад, bittorrent-клієнт *Transmission* має дві версії: *transmission-gtk* і *transmission-qt*. Який з двох слід встановити, щоб забезпечити максимальну інтеграцію з KDE?

3. Які настільні середовища Linux рекомендуються для недорогих одноплатних комп'ютерів із невеликою обчислювальною потужністю?

Дослідницькі вправи

1. Є два способи скопіювати та вставити текст у системі X Window: за допомогою традиційних клавіш `Ctrl` + `c` і `Ctrl` + `v` (також доступних у меню вікна) або натиснути середню кнопку миші, щоб вставити виділений текст. Який спосіб скопіювати та вставити текст із емулятора терміналу?

2. Більшість настільних середовищ призначають комбінацію клавіш `Alt` + `F2` для вікна *Виконати програму*, де програми можна запускати за допомогою командного рядка. У KDE, яка команда виконає типовий емулятор терміналу?

3. Який протокол найкраще підходить для доступу до віддаленого робочого столу Windows із робочого середовища Linux?

Підсумки

Цей урок є оглядом графічних робочих столів, доступних для систем Linux. Сама система X Window надає лише прості функції інтерфейсу, тому середовище робочого столу розширює можливості користувача в графічному віконному інтерфейсі. На уроці розглядались такі теми:

- Концепції графічного інтерфейсу та системи X Window.
- Настільні середовища, доступні для Linux.
- Подібності та відмінності між робочими середовищами.
- Як отримати доступ до середовища віддаленого робочого столу.

Розглянуті концепції та програми:

- Система X Window.
- Популярні робочі середовища: KDE, Gnome, Xfce.
- Протоколи віддаленого доступу: XDMCP, VNC, RDP, Spice.

Відповіді до вправ посібника

1. Який тип програми забезпечує сеанси віконної оболонки в робочому середовищі?

   Будь-який емулятор терміналу, наприклад, Konsole, термінал Gnome, xterm тощо, надасть доступ до локальної інтерактивної сесії оболонки.

2. Через різноманітність настільних середовищ Linux одна і та сама програма може мати більше однієї версії, кожна з яких найкраще підходить для певного інструментарію віджетів. Наприклад, bittorrent-клієнт *Transmission* має дві версії: *transmission-gtk* і *transmission-qt*. Який з двох слід встановити, щоб забезпечити максимальну інтеграцію з KDE?

   KDE побудовано на основі бібліотеки Qt, тому слід встановити версію Qt — *transmission-qt*.

3. Які настільні середовища Linux рекомендуються для недорогих одноплатних комп'ютерів із невеликою обчислювальною потужністю?

   Базові робочі середовища, які не використовують занадто багато візуальних ефектів, наприклад Xfce і LXDE.

Відповіді до дослідницьких вправ

1. Є два способи скопіювати та вставити текст у системі X Window: за допомогою традиційних клавіш `Ctrl` + `c` і `Ctrl` + `v` (також доступних у меню вікна) або натиснути середню кнопку миші, щоб вставити виділений текст. Який спосіб скопіювати та вставити текст із емулятора терміналу?

   Інтерактивні сеанси оболонки призначають натискання клавіші `Ctrl` + `c` для зупинки виконання програми, тому рекомендується використовувати метод середньої кнопки.

2. Більшість настільних середовищ призначають комбінацію клавіш `Alt` + `F2` для вікна *Виконати програму*, де програми можна запускати за допомогою командного рядка. У KDE, яка команда виконає типовий емулятор терміналу?

   Команда `konsole` запускає емулятор терміналу KDE, але такі загальні команди, як *terminal*, також працюють.

3. Який протокол найкраще підходить для доступу до віддаленого робочого столу Windows із робочого середовища Linux?

   Remote Desktop Protocol (RDP), оскільки він нативно підтримується Windows і Linux.

106.3 Доступність

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 106.3

Ваговий коефіцієнт

1

Ключові галузі знань

- Базові знання візуальних налаштувань і тем.
- Базові знання допоміжних технологій.

Частковий список файлів, термінів та утиліт, що використовуються

- Висококонтрастні/великі теми для робочого столу.
- Програма читання з екрану.
- Брайлівський дисплей.
- Екранна лупа.
- Екранна клавіатура.
- Залипання/повторне натискання клавіш.
- Повільні/випадкові клавіші, клавіші-перемикачі.
- Клавіші мишки.
- Жести.
- Розпізнавання голосу.

106.3 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	106 Інтерфейси користувача та робочі столи
Тема:	106.3 Доступність
Урок:	1 з 1

Вступ

Середовище робочого столу Linux має багато налаштувань та інструментів для адаптації інтерфейсу користувача для людей з обмеженими можливостями. Звичайні пристрої інтерфейсу — екран, клавіатуру та мишу/сенсорну панель — можна індивідуально налаштувати для подолання порушень зору або обмеженої рухливості.

Можна, наприклад, налаштувати колірну схему робочого столу для кращого сприйняття користувачами, які мають проблему дальтонізму. Крім того, травмовані люди можуть скористатися альтернативними методами друку та вказівки.

Деякі з цих спеціальних можливостей надаються самим середовищем робочого столу, наприклад Gnome або KDE, а інші надаються додатковими програмами. В останньому випадку важливо вибрати інструмент, який найкраще інтегрується з середовищем робочого столу, щоб допомога була кращою.

Налаштування доступності

Усі основні дистрибутиви Linux забезпечують приблизно однакові функції доступності, які

можна налаштувати за допомогою модуля конфігурації. Його можна знайти в менеджері налаштувань, який постачається разом із середовищем робочого столу. Модуль налаштувань доступності називається *Універсальний доступ* у робочому столі Gnome, тоді як у KDE знаходиться в розділі *Системні налаштування*, *Персоналізація*, *Доступність*. В інших робочих середовищах, наприклад *Xfce*, присутня така ж назва — *Доступність* у диспетчері графічних налаштувань. Однак вони пропонують менший набір функцій порівняно з Gnome і KDE.

Gnome, наприклад, можна налаштувати на постійне відображення меню універсального доступу у верхньому правому куті екрана, де можна швидко ввімкнути параметри доступності. Його можна використовувати, наприклад, для заміни звукового сповіщення на візуальне, щоб користувачі з вадами слуху легше сприймали системні сповіщення. Хоча KDE не має подібного меню швидкого доступу, функція візуального сповіщення також доступна, але замість цього вона називається *візуальний дзвіночок*.

Помічник для налаштування клавіатури та миші

Поведінку клавіатури та миші за замовчуванням можна змінити, щоб уникнути певних проблем із мобільністю. Комбінації клавіш, частота автоматичного повторення клавіш і ненавмисне натискання клавіш можуть бути серйозними перешкодами для користувачів із обмеженою рухливістю рук. Ці недоліки введення тексту усуваються трьома функціями доступності, пов'язаними з клавіатурою: *Липкі клавіші*, *Тремтячі клавіші* і *Повільні клавіші*.

Функція *Липкі клавіші* (Sticky keys), яка знаходиться в розділі *Помічник при наборі тексту* конфігурації універсального доступу Gnome, дозволяє користувачеві вводити комбінації клавіш по одній клавіші за раз. Якщо ця функція ввімкнена, комбінації клавіш, наприклад, `Ctrl` + `C`, не потрібно утримувати одночасно. Користувач може спочатку натиснути клавішу `Ctrl`, відпустити її, а потім натиснути клавішу `C`. У KDE цей параметр знаходиться на вкладці *Клавіші-модифікатори* у вікні налаштувань спеціальних можливостей. KDE також пропонує опцію *блокування клавіш*: якщо її ввімкнути, клавіші `Alt`, `Ctrl` і Shift залишатимуться "натиснутими", якщо користувач натисне їх двічі, подібно до поведінки клавіші Caps Lock. Подібно до функції Caps Lock, користувачеві потрібно буде знову натиснути відповідну клавішу, щоб відпустити її.

Функція *Тремтячі клавіші* (Bounce keys) намагається запобігти ненавмисним натисканням клавіш, встановлюючи затримку між ними, тобто нове натискання клавіш буде прийнято лише після того, як мине певний проміжок часу з моменту останнього натискання клавіші. Користувачам із тремтінням рук може бути корисною функція відскоку клавіш, щоб уникнути багаторазового натискання клавіші, коли вони мають намір натиснути її лише один раз. У Gnome ця функція стосується лише повторення тих самих клавіш, тоді як

у KDE вона також стосується натискання будь-якої іншої клавіші та знаходиться на вкладці *Клавіатурні фільтри*.

Функція *Повільні клавіші* (Slow keys) також допомагає уникнути випадкового натискання клавіш. Якщо функція ввімкнена, повільні клавіші вимагатимуть від користувача утримувати клавішу певний час, перш ніж її буде прийнято. Залежно від потреб користувача також може бути корисним налаштувати автоматичне повторення при утриманні клавіші, доступне в налаштуваннях клавіатури.

Функції спеціальних можливостей залипання клавіш і повільних клавіш можна вмикати та вимикати за допомогою *Жестів активації*, які виконуються на клавіатурі. У KDE параметр *Використовувати дії для активації залипаючих і повільних клавіш* має бути позначений, щоб увімкнути жести активації, тоді як у Gnome ця функція називається *Увімкнути за допомогою клавіатури* у вікні налаштування *Помічника при наборі з клавіатури*. Після ввімкнення жестів активації функція залипання клавіш буде активована після натискання клавіші Shift п'ять разів поспіль. Щоб увімкнути функцію повільних клавіш, потрібно утримувати клавішу Shift протягом восьми секунд поспіль.

Користувачі, яким зручніше використовувати клавіатуру замість миші або сенсорної панелі, можуть використовувати комбінації клавіш для пересування в середовищі робочого столу. Крім того, функція під назвою *Клавіші миші* дозволяє користувачеві керувати самим вказівником миші за допомогою цифрової клавіатури, яка є в повнорозмірних настільних клавіатурах і великих ноутбуках.

Цифрова клавіатура скомпонована у вигляді квадратної сітки, тому кожна цифра відповідає напрямку: `2` переміщує курсор униз, `4` переміщує курсор ліворуч, `7` переміщує курсор на північний захід тощо. За умовчанням, число `5` відповідає клацанню лівої кнопки миші.

У той час як у Gnome є лише перемикач, щоб увімкнути опцію «Клавіші миші» у вікні налаштувань універсального доступу, у KDE параметри «Клавіші миші» знаходяться в *Параметрах системи*, *Миші*, *Навігації клавіатурою*, і там можна налаштувати такі параметри, як швидкість і прискорення.

TIP

Повільні клавіші, липкі клавіші, тремтячі клавіші і клавіші миші — це функції спеціальних можливостей, надані AccessX, ресурсом у розширенні клавіатури X системи X Window. Параметри AccessX також можна змінити з командного рядка за допомогою команди `xkbset`.

Мишу або сенсорну панель можна використовувати для введення з клавіатури, коли використання клавіатури надто незручне або неможливе. Якщо перемикач *Екранна клавіатура* в налаштуваннях універсального доступу Gnome увімкнено, екранна

клавіатура з'являтиметься щоразу, коли курсор буде в текстовому полі, а новий текст буде введено клацанням клавіш мишею або сенсорним екраном, подібно до віртуальної клавіатури смартфонів.

KDE та інші настільні середовища можуть не надавати екранну клавіатуру за замовчуванням, але пакет *onboard* можна встановити вручну, щоб забезпечити просту екранну клавіатуру, яку можна використовувати в будь-якому настільному середовищі. Після встановлення він буде доступний як звичайний додаток у панелі запуску програм.

Поведінку вказівника також можна змінити, якщо клацання та перетягування миші викликає біль або є неможливим з будь-якої іншої причини. Якщо користувач не може натиснути кнопку миші досить швидко, щоб ініціювати подію подвійного клацання, наприклад, інтервал часу для повторного натискання кнопки миші для подвійного клацання можна збільшити в *Налаштуваннях миші* у системному конфігураційному вікні.

Якщо користувач не може натиснути одну з кнопок миші або жодну з кнопок миші, тоді клацання миші можна імітувати за допомогою різних технік. У розділі *Допомога при клацанні Універсального доступу Gnome* параметр *Імітувати клацання правою кнопкою миші* генеруватиме клацання правою кнопкою миші, якщо користувач натискає та утримує ліву кнопку миші. Якщо ввімкнено параметр *Імітувати клацання шляхом наведення курсора*, подія клацання буде ініційована, коли користувач нерухомо тримає мишу. У KDE програма *KMouseTool* надасть ті самі можливості для допомоги у діях миші.

Порушення зору

Користувачі зі зниженим зором все ж можуть використовувати екран монітора для взаємодії з комп'ютером. Залежно від потреб користувача можна зробити багато візуальних налаштувань, щоб покращити деталі стандартного графічного робочого столу, які інакше важко побачити.

Розділ Gnome *Seeing* налаштувань *Універсальний доступ* надає параметри, які можуть допомогти людям із вадами зору:

Висока контрастність

полегшить перегляд вікон і кнопок, намалювавши їх більш чіткими кольорами.

Великий текст

збільшить стандартний розмір екранного шрифту.

Розмір курсора

дозволяє вибрати більший курсор миші, що полегшує його пошук на екрані.

Деякі з цих налаштувань не пов'язані виключно зі спеціальними можливостями, тому їх можна знайти в розділі зовнішнього вигляду утиліти налаштування, що надається в інших середовищах робочого столу. Користувач, якому важко розрізнити візуальні елементи, може вибрати висококонтрастну тему, щоб було легше ідентифікувати кнопки, вікна, що перекриваються, тощо.

Якщо одних лише налаштувань зовнішнього вигляду недостатньо для покращення читабельності екрана, то для збільшення частин екрана можна використати програму екранної лупи. Ця функція називається *Масштаб* у налаштуваннях *Універсальний доступ* робочого середовища Gnome, де можна налаштувати такі параметри, як коефіцієнт збільшення, положення лупи та налаштування кольору.

У KDE програма *KMagnifier* надає ті самі функції, але вона доступна як звичайна програма через засіб запуску програм. Інші робочі середовища можуть надавати власні екранні лупи. Xfce, наприклад, буде збільшувати та зменшувати масштаб екрана, обертаючи колесо прокручування миші, коли клавіша `Alt` натиснута.

Нарешті, користувачі, для яких графічний інтерфейс недоступний, можуть використовувати *зчитувач з екрана* для взаємодії з комп'ютером. Незалежно від обраного середовища робочого столу, найпопулярнішим програмою зчитування екрана для систем Linux є *Orca*, яка зазвичай встановлюється за замовчуванням у більшості дистрибутивів. Orca генерує синтезований голос, щоб повідомляти про події на екрані та читати текст під курсором миші. Orca також працює з *оновлюваними дисплеями Брайля*, спеціальними пристроями, які відображають символи Брайля, піднімаючи маленькі шпильки, які можна відчути кінчиками пальців. Не всі настільні програми повністю адаптовані для програм зчитування з екрана, і не всім користувачам буде легко ними користуватися, тому важливо надати якомога більше стратегій зчитування з екрана для користувачів на вибір.

Вправи до посібника

1. Яка функція доступності може допомогти користувачеві перемикатися між відкритими вікнами за допомогою клавіатури, враховуючи, що користувач не може натиснути клавіші `Alt` і `Tab` одночасно?

2. Як функція доступності *Тремтячі клавіші* може допомогти користувачам, яким мимовільне тремтіння рук заважає набирати текст?

3. Які найпоширеніші жести активації для функції спеціальних можливостей *Липкі клавіші*?

Дослідницькі вправи

1. Функції доступності можуть не надаватися окремою програмою та можуть відрізнятися від одного настільного середовища до іншого. Яка програма в KDE допомагає тим, хто має травми, клацаючи мишкою щоразу, коли курсор миші на короткий час зупиняється?

2. Які аспекти зовнішнього вигляду графічного середовища можна змінити, щоб людям було легше читати текст на екрані?

3. Яким чином програма *Orca* може допомогти користувачам із вадами зору взаємодіяти з настільним середовищем?

Підсумки

Цей урок охоплює загальні функції доступності, доступні в системах Linux. Усі основні робочі середовища, особливо Gnome і KDE, містять багато вбудованих і сторонніх програм, які допомагають людям із вадами зору або обмеженою рухливістю. На уроці розглядаються такі теми:

- Як змінити параметри доступності.
- Альтернативні способи використання клавіатури та миші.
- Покращення робочого столу для людей із вадами зору.

Розглянуті команди та процедури:

- Налаштування спеціальних можливостей клавіатури: липкі клавіші, повільні клавіші, тремтячі клавіші.
- Штучне генерування подій миші.
- Екранна клавіатура.
- Візуальні налаштування для покращення читабельності.
- Висококонтрастні/великі теми для робочого столу.
- Екранні лупи.
- Програма зчитування екрана Orca.

Відповіді до вправ посібника

1. Яка функція доступності може допомогти користувачеві перемикатися між відкритими вікнами за допомогою клавіатури, враховуючи, що користувач не може натиснути клавіші `Alt` і `Tab` одночасно?

   Функція Повільні клавіші, яка дозволяє користувачеві вводити комбінації клавіш по одній клавіші за раз.

2. Як функція доступності *Тремтячі клавіші* може допомогти користувачам, яким мимовільне тремтіння рук заважає набирати текст?

   Якщо ввімкнуто функцію Тремтячі клавіші, нове натискання клавіш буде прийнято лише після того, як мине певний проміжок часу з моменту останнього натискання клавіші.

3. Які найпоширеніші жести активації для функції спеціальних можливостей *Липкі клавіші*?

   Якщо активовані жести активації, функція залипання клавіш буде активована після натискання клавіші `Shift` п’ять разів поспіль.

Відповіді до дослідницьких вправ

1. Функції доступності можуть не надаватися окремою програмою та можуть відрізнятися від одного настільного середовища до іншого. Яка програма в KDE допомагає тим, хто має травми, клацаючи мишкою щоразу, коли курсор миші на короткий час зупиняється?

   Застосунок *KMouseTool*.

2. Які аспекти зовнішнього вигляду графічного середовища можна змінити, щоб людям було легше читати текст на екрані?

   Установлення великого розміру екранного шрифту в конфігурації робочого столу полегшить читання всіх екранних текстів.

3. Яким чином програма *Orca* може допомогти користувачам із вадами зору взаємодіяти з настільним середовищем?

   Orca — це програма зчитування з екрана, яка генерує синтезований голос, щоб повідомляти про події на екрані та читати текст під курсором миші. Вона також працює з пристроями під назвою *оновлювані дисплеї Брайля*, тож користувач може ідентифікувати текст за допомогою тактильних шаблонів.

Linux Professional Institute

Розділ 107: Адміністративні завдання

107.1 Керування обліковими записами користувачів і груп і відповідними системними файлами

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 107.1

Ваговий коефіцієнт

5

Ключові галузі знань

- Додавання, змінення та видалення користувачів і груп.
- Керування інформацією про користувача/групу в базах даних паролів/груп.
- Створення та керування спеціальними та обмеженими обліковими записами.

Частковий список файлів, термінів та утиліт, що використовуються

- `/etc/passwd`
- `/etc/shadow`
- `/etc/group`
- `/etc/skel/`
- `chage`
- `getent`
- `groupadd`
- `groupdel`
- `groupmod`
- `passwd`

- `useradd`
- `userdel`
- `usermod`

107.1 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	107 Адміністративні завдання
Тема:	107.1 Керування обліковими записами користувачів і груп і відповідними системними файлами
Урок:	1 з 2

Вступ

Адміністрування користувачів і груп є дуже важливою частиною роботи будь-якого системного адміністратора. Сучасні дистрибутиви Linux реалізують графічні інтерфейси, які дозволяють швидко й легко керувати всіма діями, пов'язаними з цим ключовим аспектом. Такі інтерфейси відрізняються один від одного графічним оформленням, але функції однакові. За допомогою цих інструментів ви можете переглядати, редагувати, додавати та видаляти локальних користувачів і групи. Однак для більш розширеного керування вам потрібно працювати через командний рядок.

Додавання облікових записів користувачів

У Linux ви можете додати новий обліковий запис користувача за допомогою команди `useradd`. Наприклад, діючи з привілеями root, ви можете створити новий обліковий запис користувача з іменем `michael` із налаштуванням за замовчуванням, використовуючи наступне:

```
# useradd michael
```

Коли ви виконуєте команду `useradd`, інформація про користувачів та групи, що зберігається в базах даних паролів і груп, оновлюється для новоствореного облікового запису користувача, а також, якщо вказано, створюється домашній каталог нового користувача. Також створюється група з іменем нового облікового запису користувача.

Після створення нового користувача ви можете встановити його пароль за допомогою команди `passwd`. Ви можете переглянути його ідентифікатор користувача (UID), ідентифікатор групи (GID) і груп, до яких він належить, за допомогою команд `id` і `groups`.

```
# passwd michael
Changing password for user michael.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
# id michael
uid=1000(michael) gid=100(michael) groups=100(michael)
# groups michael
michael : michael
```

NOTE

Пам’ятайте, що будь-який користувач може переглянути свій UID, GID і групи, до яких він належить, просто використовуючи команди `id` і `groups` без аргументів, і що будь-який користувач може змінити свій пароль за допомогою команди `passwd`. Однак лише користувачі з привілеями root можуть змінити пароль *будь-якого* користувача.

Найважливіші параметри, які застосовуються до команди `useradd`:

`-c`

створити новий обліковий запис користувача з власними коментарями (наприклад, повне ім’я користувача).

`-d`

створити новий обліковий запис користувача з власним домашнім каталогом.

`-e`

створити новий обліковий запис користувача, встановивши конкретну дату, коли його буде вимкнено.

`-f`

створити новий обліковий запис користувача, встановивши кількість днів після закінчення терміну дії пароля, протягом якого користувач повинен оновити пароль (інакше обліковий запис буде вимкнено).

`-g`

створити новий обліковий запис користувача з певним GID.

`-G`

створити новий обліковий запис користувача, додавши його до кількох вторинних груп.

`-k`

створити новий обліковий запис користувача, скопіювавши шаблонні файли з певного спеціального каталогу (цей параметр дійсний, лише якщо вказано параметр `-m` або `--create-home`).

`-m`

створити новий обліковий запис користувача з його домашнім каталогом (якщо він не існує).

`-M`

створити новий обліковий запис користувача без домашнього каталогу.

`-s`

створити новий обліковий запис користувача з певною оболонкою входу.

`-u`

створити новий обліковий запис користувача з певним UID.

Перегляньте man-сторінки для команди `useradd`, щоб отримати повний список параметрів.

Змінення облікових записів користувачів

Іноді вам потрібно змінити атрибут існуючого облікового запису користувача, наприклад ім'я для входу, оболонку для входу, термін дії пароля тощо. У таких випадках вам потрібно використовувати команду `usermod`.

```
# usermod -s /bin/tcsh michael
```

```
# usermod -c "Michael User Account" michael
```

Так само, як і для команди `useradd`, для команди `usermod` потрібні привілеї root.

У наведених вище прикладах спочатку змінено оболонку входу `michael`, а потім до цього облікового запису користувача додано короткий опис. Пам'ятайте, що ви можете змінити кілька атрибутів одночасно, вказавши їх в одній команді.

Найважливіші параметри, які застосовуються до команди `usermod`:

`-c`

додати короткий коментар до вказаного облікового запису користувача.

`-d`

змінити домашній каталог указаного облікового запису користувача. Якщо використовується з опцією `-m`, вміст поточного домашнього каталогу переміщується до нового домашнього каталогу, який створюється, якщо він ще не існує.

`-e`

встановити дату закінчення терміну дії вказаного облікового запису користувача.

`-f`

встановити кількість днів після закінчення терміну дії пароля, протягом якого користувач повинен оновити пароль (інакше обліковий запис буде вимкнено).

`-g`

змінити основну групу вказаного облікового запису користувача (група має існувати).

`-G`

додати вторинні групи до вказаного облікового запису користувача. Кожна група має існувати та відділятися від наступної комою без пробілів. Якщо використовується окремо, цей параметр видаляє всі існуючі групи, до яких належить користувач, тоді як, якщо використовується з параметром `-a`, він просто додає нові вторинні групи до існуючих.

`-l`

змінити ім'я для входу вказаного облікового запису користувача.

`-L`

заблокувати вказаний обліковий запис користувача. Це ставить знак оклику перед зашифрованим паролем у файлі `/etc/shadow`, таким чином вимикаючи доступ із

паролем для цього користувача.

`-s`

змінити оболонку входу вказаного облікового запису користувача.

`-u`

змінити UID зазначеного облікового запису користувача.

`-U`

розблокувати вказаний обліковий запис користувача. Це видаляє знак оклику перед зашифрованим паролем у файлі `/etc/shadow`.

Перегляньте man-сторінки для команди `usermod`, щоб отримати повний список параметрів.

TIP

Пам'ятайте, що коли ви змінюєте ім'я для входу в обліковий запис користувача, вам, імовірно, слід перейменувати домашній каталог цього користувача та інші елементи, пов'язані з користувачем, наприклад файли буфера електронної пошти. Також пам'ятайте, що коли ви змінюєте UID облікового запису користувача, вам, імовірно, слід виправити право власності на файли та каталоги за межами домашнього каталогу користувача (ідентифікатор користувача змінюється автоматично для поштової скриньки користувача та для всіх файлів, які належать користувачу та розташовані в домашньому каталозі користувача).

Видалення облікових записів користувачів

Якщо ви хочете видалити обліковий запис користувача, ви можете скористатися командою `userdel`. Зокрема, ця команда оновлює інформацію, що зберігається в базах даних облікових записів, видаляючи всі записи, що стосуються зазначеного користувача. Параметр `-r` також видаляє домашній каталог користувача та весь його вміст разом із буфером електронної пошти користувача. Інші файли, розташовані в інших місцях, потрібно шукати та видаляти вручну.

```
# userdel -r michael
```

Так само, як для `useradd` і `usermod`, для видалення облікових записів користувачів потрібні права root.

Додавання, зміна та видалення груп

Подібно до керування користувачами, ви можете додавати, змінювати та видаляти групи за допомогою команд `groupadd`, `groupmod` і `groupdel` з правами root. Якщо ви хочете створити нову групу під назвою `developer`, ви можете виконати таку команду:

```
# groupadd -g 1090 developer
```

Опція `-g` цієї команди створює групу з певним GID.

WARNING Пам'ятайте, що коли ви додаєте новий обліковий запис користувача, первинна група та вторинні групи, до яких він належить, *повинні* існувати до запуску команди `useradd`.

Пізніше, якщо ви захочете перейменувати групу з `developer` на `web-developer` і змінити її GID, ви можете виконати наступну команду:

```
# groupmod -n web-developer -g 1050 developer
```

TIP Пам'ятайте, що якщо ви змінюєте GID за допомогою параметра `-g`, ви повинні змінити GID усіх файлів і каталогів, які мають належати до цієї групи.

Нарешті, якщо ви хочете видалити групу `web-developer`, ви можете виконати наступне:

```
# groupdel web-developer
```

Ви не можете видалити групу, якщо вона є основною групою облікового запису користувача. Тому ви повинні видалити користувача перед видаленням групи. Що стосується користувачів, якщо ви видаляєте групу, файли, що належать до цієї групи, залишаються у вашій файловій системі й не видаляються та не призначаються іншій групі.

Шаблонний каталог

Коли ви додаєте новий обліковий запис користувача, навіть створюючи його домашній каталог, новостворений домашній каталог заповнюється файлами та папками, які копіюються з шаблонного каталогу (за замовчуванням `/etc/skel`). Ідея цього проста: системний адміністратор хоче додати нових користувачів, які мають ті самі файли та каталоги у своїй домашній папці. Таким чином, якщо ви хочете налаштувати файли та

папки, які автоматично створюються в домашньому каталозі нових облікових записів користувачів, ви повинні додати ці нові файли та папки до шаблонного каталогу.

TIP Зауважте, що якщо ви хочете отримати список усіх файлів і каталогів у шаблонному каталозі, ви повинні використати команду `ls -al`.

Файл `/etc/login.defs`

У Linux файл `/etc/login.defs` визначає параметри конфігурації, які керують створенням користувачів і груп. Крім того, команди, показані в попередніх розділах, приймають значення за замовчуванням із цього файлу.

Найважливіші директиви:

`UID_MIN` і `UID_MAX`

Діапазон ідентифікаторів користувачів, які можна призначити новим звичайним користувачам.

`GID_MIN` і `GID_MAX`

Діапазон ідентифікаторів груп, які можна призначити новим звичайним групам.

`CREATE_HOME`

Вказує, чи слід створювати домашній каталог за замовчуванням для нових користувачів.

`USERGROUPS_ENAB`

Вказує, чи повинна система за замовчуванням створювати нову групу для кожного нового облікового запису користувача з тим же іменем, що й у користувача, і чи видалення облікового запису користувача також має видаляти основну групу користувача, якщо вона більше не містить учасників.

`MAIL_DIR`

Каталог буфера електронної пошти.

`PASS_MAX_DAYS`

Максимальна кількість днів, протягом яких можна використовувати пароль.

`PASS_MIN_DAYS`

Мінімальна кількість днів між змінами пароля.

`PASS_MIN_LEN`

Мінімально прийнятна довжина пароля.

`PASS_WARN_AGE`

Кількість днів попередження до закінчення терміну дії пароля.

TIP Під час керування користувачами та групами завжди перевіряйте цей файл, щоб переглянути та, зрештою, змінити типову поведінку системи, якщо це необхідно.

Команда `passwd`

Ця команда в основному використовується для зміни пароля користувача. Як описано раніше, будь-який користувач може змінити свій власний пароль, але лише root може змінити пароль *будь-якого* користувача. Це відбувається тому, що команда `passwd` має встановлений біт SUID (`s` замість позначки виконуваного файлу для власника). Це означає, що вона виконується з привілеями власника файлу (отже, root).

```
# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 42096 mag 17  2015 /usr/bin/passwd
```

Залежно від параметрів `passwd`, які використовуються, ви можете контролювати певні аспекти старіння пароля:

`-d`

Видалити пароль облікового запису користувача (таким чином вимкнувши користувача).

`-e`

Змусити змінити пароль для облікового запису користувача.

`-i`

Встановити кількість днів бездіяльності після закінчення терміну дії пароля, протягом яких користувач повинен оновити пароль (інакше обліковий запис буде вимкнено).

`-l`

Блокування облікового запису користувача (зашифрований пароль має префікс знаку оклику у файлі `/etc/shadow`).

`-n`

Встановити мінімальний термін дії пароля.

`-S`

Вивести інформацію про статус пароля певного облікового запису користувача.

`-u`

Розблокувати обліковий запис користувача (знак оклику видалено з поля пароля у файлі `/etc/shadow`).

`-x`

Встановити максимальний термін дії пароля.

`-w`

Встановити кількість днів попередження до закінчення терміну дії пароля, протягом яких користувач буде попереджений про необхідність змінити пароль.

NOTE

Групи також можуть мати пароль, який можна встановити за допомогою команди `gpasswd`. Користувачі, які не є членами групи, але знають її пароль, можуть тимчасово приєднатися до неї за допомогою команди `newgrp`. Пам'ятайте, що `gpasswd` також використовується для додавання та видалення користувачів із групи та встановлення списку адміністраторів і звичайних учасників групи.

Команда `chage`

Ця команда, яка розшифровується як "change age", використовується для зміни інформації про старість пароля користувача. Команда `chage` обмежена для користувача root, за винятком опції `-l`, яку можуть використовувати звичайні користувачі для перегляду інформації про старіння пароля свого облікового запису.

Інші параметри, які застосовуються до команди `chage`:

`-d`

Встановити останню зміну пароля для облікового запису користувача.

`-E`

Встановити дату закінчення терміну дії облікового запису користувача.

`-I`

Встановити кількість днів бездіяльності після закінчення терміну дії пароля, протягом

яких користувач повинен оновити пароль (інакше обліковий запис буде вимкнено).

`-m`

Встановити мінімальний термін дії пароля для облікового запису користувача.

`-M`

Встановити максимальний термін дії пароля для облікового запису користувача.

`-W`

Встановити кількість днів попередження до закінчення терміну дії пароля, протягом яких користувач буде попереджений про необхідність змінити пароль.

Вправи до посібника

1. Для кожної з наступних команд визначте відповідну мету:

`usermod -L`	
`passwd -u`	
`chage -E`	
`groupdel`	
`useradd -s`	
`groupadd -g`	
`userdel -r`	
`usermod -l`	
`groupmod -n`	
`useradd -m`	

2. Для кожної з наступних команд `passwd` визначте відповідну команду `chage`:

`passwd -n`	
`passwd -x`	
`passwd -w`	
`passwd -i`	
`passwd -S`	

3. Поясніть детально призначення команд у попередньому питанні:

4. Якими командами можна заблокувати обліковий запис користувача? І якими командами його розблокувати?

Дослідницькі вправи

1. За допомогою команди `groupadd` створіть групи `administrators` і `developers`. Припустимо, що ви працюєте як root.

2. Тепер, коли ви створили ці групи, виконайте таку команду: `useradd -G administrators,developers kevin`. Які операції виконує ця команда? Припустімо, що для `CREATE_HOME` і `USERGROUPS_ENAB` у `/etc/login.defs` встановлено значення `yes`.

3. Створіть нову групу під назвою `designers`, перейменуйте її на `web-designers` і додайте цю нову групу до вторинних груп облікового запису користувача `kevin`. Визначте всі групи, до яких належить `kevin`, та їхні ідентифікатори.

4. Видаліть лише групу `developers` із вторинних груп `kevin`.

5. Встановіть пароль для облікового запису користувача `kevin`.

6. Використовуючи команду `chage`, спочатку перевірте термін дії облікового запису користувача `kevin`, а потім змініть його на 31 грудня 2022 року. За допомогою якої іншої команди можна змінити дату закінчення терміну дії облікового запису користувача?

7. Додайте новий обліковий запис користувача під назвою `emma` з UID 1050 і встановіть `адміністратори` як основну групу, а `розробники` та `веб-дизайнери` як вторинні групи.

8. Змініть оболонку входу `emma` на `/bin/sh`.

9. Видаліть облікові записи користувачів `emma` і `kevin`, а також групи `administrators`, `developers` і `web-designers`.

Підсумки

На цьому уроці ви дізналися:

- Основи керування користувачами та групами в Linux.
- Як додавати, змінювати та видаляти облікові записи користувачів.
- Як додавати, змінювати та видаляти облікові записи групи.
- Підтримка шаблонного каталогу.
- Редагування файлу, який керує створенням користувачів і груп.
- Зміна паролів облікових записів користувачів.
- Зміна інформації про старіння пароля облікових записів користувачів.

У цьому уроці обговорювалися такі файли та команди:

`useradd`
: Створити новий обліковий запис користувача.

`usermod`
: Змінити обліковий запис користувача.

`userdel`
: Видалити обліковий запис користувача.

`groupadd`
: Створити новий обліковий запис групи.

`groupmod`
: Змінити обліковий запис групи.

`groupdel`
: Видалити обліковий запис групи.

`passwd`
: Змінити пароль облікових записів користувачів і контролювати всі аспекти старіння пароля.

`chage`
: Змінити інформацію про термін дії пароля користувача.

`/etc/skel`

Стандартне розташування шаблонного каталогу.

`/etc/login.defs`

Файл, який керує створенням користувачів і груп і надає значення за замовчуванням для деяки параметрів облікового запису користувача.

Відповіді до вправ посібника

1. Для кожної з наступних команд визначте відповідну мету:

`usermod -L`	Заблокувати обліковий запис користувача
`passwd -u`	Розблокувати обліковий запис користувача
`chage -E`	Встановити термін дії облікового запису користувача
`groupdel`	Видалити групу
`useradd -s`	Створити новий обліковий запис користувача зі спеціальною оболонкою входу
`groupadd -g`	Створити нову групу з певним GID
`userdel -r`	Видалити обліковий запис користувача та всі файли в його домашньому каталозі, сам домашній каталог та буферні файли електронної пошти користувача
`usermod -l`	Змінити ім’я для входу в обліковий запис користувача
`groupmod -n`	Змінити назву групи
`useradd -m`	Створити новий обліковий запис користувача та його домашній каталог

2. Для кожної з наступних команд `passwd` визначте відповідну команду `chage`:

`passwd -n`	`chage -m`
`passwd -x`	`chage -M`
`passwd -w`	`chage -W`
`passwd -i`	`chage -I`
`passwd -S`	`chage -l`

3. Поясніть детально призначення команд у попередньому питанні:

   У Linux ви можете використовувати команду `passwd -n` (або `chage -m`), щоб встановити

мінімальну кількість днів між змінами пароля, команду `passwd -x` (або `chage -M`), щоб встановити максимальну кількість днів, протягом яких пароль дійсний, команду `passwd -w` (або `chage -W`), щоб встановити кількість днів попередження до закінчення терміну дії пароля, команду `passwd -i` (або `chage -I`), щоб встановити кількість днів бездіяльності, протягом яких користувач повинен змінити пароль, і команду `passwd -S` (або `chage -l`), щоб показати коротку інформацію про пароль облікового запису користувача.

4. Якими командами можна заблокувати обліковий запис користувача? І якими командами його розблокувати?

   Якщо ви хочете заблокувати обліковий запис користувача, ви можете скористатися однією з цих команд: `usermod -L`, `usermod --lock` і `passwd -l`. Натомість, якщо ви хочете його розблокувати, ви можете використати `usermod -U`, `usermod --unlock` і `passwd -u`.

Відповіді до дослідницьких вправ

1. За допомогою команди `groupadd` створіть групи `administrators` і `developers`. Припустимо, що ви працюєте як root.

```
# groupadd administrators
# groupadd developers
```

2. Тепер, коли ви створили ці групи, виконайте таку команду: `useradd -G administrators,developers kevin`. Які операції виконує ця команда? Припустімо, що для `CREATE_HOME` і `USERGROUPS_ENAB` у `/etc/login.defs` встановлено значення `yes`.

   Команда додає нового користувача з іменем `kevin` до списку користувачів у системі, створює його домашній каталог (`CREATE_HOME` має значення yes, тому ви можете опустити параметр -m) і створює нову групу з назвою `kevin` як основну групу цього облікового запису користувача (`USERGROUPS_ENAB` має значення yes). Нарешті, файли та папки, що містяться в шаблонному каталозі, копіюються до домашнього каталогу `kevin`.

3. Створіть нову групу під назвою `designers`, перейменуйте її на `web-designers` і додайте цю нову групу до вторинних груп облікового запису користувача `kevin`. Визначте всі групи, до яких належить `kevin`, та їхні ідентифікатори.

```
# groupadd designers
# groupmod -n web-designers designers
# usermod -a -G web-designers kevin
# id kevin
uid=1010(kevin) gid=1030(kevin)
groups=1030(kevin),1028(administrators),1029(developers),1031(web-designers)
```

4. Видаліть лише групу `developers` із вторинних груп `kevin`.

```
# usermod -G administrators,web-designers kevin
# id kevin
uid=1010(kevin) gid=1030(kevin) groups=1030(kevin),1028(administrators),1031(web-
designers)
```

   Команда `usermod` не має можливості видалити лише одну групу; тому вам потрібно вказати всі вторинні групи, до яких належить користувач.

5. Встановіть пароль для облікового запису користувача `kevin`.

```
# passwd kevin
Changing password for user kevin.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

6. Використовуючи команду `chage`, спочатку перевірте термін дії облікового запису користувача `kevin`, а потім змініть його на 31 грудня 2022 року. За допомогою якої іншої команди можна змінити дату закінчення терміну дії облікового запису користувача?

```
# chage -l kevin | grep "Account expires"
Account expires     : never
# chage -E 2022-12-31 kevin
# chage -l kevin | grep "Account expires"
Account expires     : dec 31, 2022
```

   Команда `usermod` з опцією `-e` еквівалентна `chage -E`.

7. Додайте новий обліковий запис користувача під назвою `emma` з UID 1050 і встановіть `administrators` як основну групу, а `developers` та `web-designers` як вторинні групи.

```
# useradd -u 1050 -g administrators -G developers,web-designers emma
# id emma
uid=1050(emma) gid=1028(administrators)
groups=1028(administrators),1029(developers),1031(web-designers)
```

8. Змініть оболонку входу `emma` на `/bin/sh`.

```
# usermod -s /bin/sh emma
```

9. Видаліть облікові записи користувачів `emma` і `kevin`, а також групи `administrators`, `developers` і `web-designers`.

```
# userdel -r emma
# userdel -r kevin
# groupdel administrators
# groupdel developers
```

```
# groupdel web-designers
```

107.1 Урок 2

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	107 Адміністративні завдання
Тема:	107.1 Керування обліковими записами користувачів і груп і відповідними системними файлами
Урок:	2 з 2

Вступ

Інструменти командного рядка, розглянуті в попередньому уроці, і графічні програми, що надаються кожним дистрибутивом, які виконують однакові завдання, оновлюють ряд файлів, які зберігають інформацію про користувачів і групи.

Ці файли розташовані в каталозі `/etc/`, а саме:

`/etc/passwd`
: Файл із семи полів, розділених двокрапками, що містить основну інформацію про користувачів.

`/etc/group`
: Файл із чотирма полями, розділеними двокрапками, що містить основну інформацію про групи.

`/etc/shadow`

Файл з дев'яти полів, розділених двокрапками, що містить зашифровані паролі користувачів.

`/etc/gshadow`

Файл із чотирма полями, розділеними двокрапками, що містить зашифровані паролі груп.

Хоча ці чотири файли є звичайним текстом, їх не слід редагувати безпосередньо, а завжди за допомогою інструментів, наданих дистрибутивом, який ви використовуєте.

/etc/passwd

Це доступний для читання файл, який містить список користувачів, кожен в окремому рядку. Кожен рядок складається із семи полів, розділених двокрапками:

Ім'я користувача

Ім'я, яке використовується під час входу користувача в систему.

Пароль

Зашифрований пароль (або `x`, якщо використовуються тіньові паролі).

ID користувача (UID)

Ідентифікаційний номер, присвоєний користувачеві в системі.

ID групи (GID)

Номер основної групи користувача в системі.

GECOS

Додаткове поле коментаря, яке використовується для додавання додаткової інформації про користувача (наприклад, повне ім'я). Поле може містити кілька записів, розділених комами.

Домашній каталог

Абсолютний шлях до домашнього каталогу користувача.

Оболонка

Абсолютний шлях до програми, яка автоматично запускається, коли користувач входить до системи (зазвичай це інтерактивна оболонка, наприклад `/bin/bash`).

/etc/group

Це доступний для читання файл, який містить список груп, кожна в окремому рядку. Кожен рядок складається з чотирьох полів, розділених двокрапками:

Назва групи

Назва групи.

Пароль групи

Зашифрований пароль групи (або `x`, якщо використовуються тіньові паролі).

ID групи (GID): Ідентифікаційний номер, присвоєний групі в системі.

Список користувачів

Розділений комами список користувачів, які належать до групи, за винятком тих, для яких це основна група.

/etc/shadow

Це файл, доступний для читання лише адміністратору та користувачам із привілеями root, який містить зашифровані паролі користувачів, кожен в окремому рядку. Кожен рядок складається з дев'яти полів, розділених двокрапками:

Ім'я користувача

Ім'я, яке використовується під час входу користувача в систему.

Зашифрований пароль

Зашифрований пароль користувача (якщо значення починається з `!`, обліковий запис заблоковано).

Дата останньої зміни пароля

Дата останньої зміни пароля, вказується як кількість днів з 01/01/1970 (значення 0 означає, що користувач повинен змінити пароль під час наступного входу).

Мінімальний вік пароля

Мінімальна кількість днів після зміни пароля, яка має пройти, перш ніж користувачеві буде дозволено змінити пароль знову.

Максимальний вік пароля

Максимальна кількість днів, яка має пройти, перш ніж потрібно змінити пароль.

Період попередження про пароль

Кількість днів до закінчення терміну дії пароля, протягом яких користувач отримує попередження про необхідність зміни пароля.

Період неактивності пароля

Кількість днів після закінчення терміну дії пароля, протягом яких користувач повинен оновити пароль. Після цього періоду, якщо користувач не змінить пароль, обліковий запис буде відключено.

Термін дії облікового запису

Дата, виражена як кількість днів з 01.01.1970, протягом яких обліковий запис користувача буде вимкнено (пусте поле означає, що термін дії облікового запису користувача ніколи не закінчиться).

Зарезервоване поле

Поле, яке зарезервовано для майбутнього використання.

/etc/gshadow

Це файл, доступний для читання лише адміністратору та користувачам із привілеями адміністратора, який містить зашифровані паролі для груп, кожен в окремому рядку. Кожен рядок складається з чотирьох полів, розділених двокрапками:

Назва групи

Назва групи.

Зашифрований пароль

Зашифрований пароль для групи (використовується, коли користувач, який не є членом групи, хоче приєднатися до групи за допомогою команди `newgrp` — якщо пароль починається з `!`, нікому не дозволено отримати доступ до групи за допомогою `newgrp`).

Адміністратори групи

Список адміністраторів групи, розділених комами (вони можуть змінювати пароль групи та можуть додавати або видаляти учасників групи за допомогою команди `gpasswd`).

Члени групи

Список учасників групи, розділених комами.

Фільтрація бази даних паролів і груп

Дуже часто може знадобитися переглянути інформацію про користувачів і групи, що зберігається в цих чотирьох файлах, і знайти конкретні записи. Щоб виконати це завдання, ви можете скористатися командою `grep` або об'єднати `cat` і `grep`.

```
# grep emma /etc/passwd
emma:x:1020:1020:User Emma:/home/emma:/bin/bash
# cat /etc/group | grep db-admin
db-admin:x:1050:grace,frank
```

Іншим способом доступу до цих баз даних є використання команди `getent`. Загалом ця команда відображає записи з баз даних, які підтримуються бібліотеками *Name Service Switch* (NSS), і вимагає імені бази даних і ключа пошуку. Якщо ключовий аргумент не надано, відображаються всі записи у вказаній базі даних (якщо база даних не підтримує перерахування). В іншому випадку, якщо надано один або кілька ключових аргументів, база даних фільтрується відповідно.

```
# getent passwd emma
emma:x:1020:1020:User Emma:/home/emma:/bin/bash
# getent group db-admin
db-admin:x:1050:grace,frank
```

Команда `getent` не потребує прав root; вам просто потрібно мати можливість читати базу даних, з якої ви хочете отримати записи.

NOTE Пам'ятайте, що `getent` може отримати доступ лише до баз даних, налаштованих у файлі `/etc/nsswitch.conf`.

Вправи до посібника

1. Подивіться на наступні вихідні дані і дайте відповідь на такі запитання:

```
# cat /etc/passwd | grep '\(root\|mail\|catherine\|kevin\)'
root:x:0:0:root:/root:/bin/bash
mail:x:8:8:mail:/var/spool/mail:/sbin/nologin
catherine:x:1030:1025:User Chaterine:/home/catherine:/bin/bash
kevin:x:1040:1015:User Kevin:/home/kevin:/bin/bash
# cat /etc/group | grep '\(root\|mail\|db-admin\|app-developer\)'
root:x:0:
mail:x:8:
db-admin:x:1015:emma,grace
app-developer:x:1016:catherine,dave,christian
# cat /etc/shadow | grep '\(root\|mail\|catherine\|kevin\)'
root:$6$1u36Ipok$ljt8ooPMLewAhkQPf.lYgGopAB.jClTO6ljsdczxvkLPkpi/amgp.zyfAN680zrLLp2avvpd
KA0llpssdfcPppOp:18015:0:99999:7:::
mail:*:18015:0:99999:7:::
catherine:$6$ABCD25jlld14hpPthEFGnnssEWw1234yioMpliABCdef1f3478kAfhhAfgbAMjY1/BAeeAsl/FeE
dddKd12345g6kPACcik:18015:20:90:5:::
kevin:$6$DEFGabc123WrLp223fsvp0ddx3dbA7pPPc4LMaa123u6Lp02Lpvm123456pyphhh5ps012vbArL245.P
R1345kkA3Gas12P:18015:0:60:7:2::
# cat /etc/gshadow | grep '\(root\|mail\|db-admin\|app-developer\)'
root:*::
mail:*::
db-admin:!:emma:emma,grace
app-developer:!::catherine,dave,christian
```

   - Що таке ідентифікатор користувача (UID) та ідентифікатор групи (GID) для `root` і `catherine`?

   - Як називається первинна група `kevin`? Чи є інші члени цієї групи?

   - Яка оболонка встановлена для `mail`? Що це означає?

   - Хто є членами групи `app-developer`? Хто з цих учасників є адміністраторами групи, а хто звичайними?

- Який мінімальний термін дії пароля для `catherine`? І який максимальний термін служби пароля?

- Який період неактивності пароля для `kevin`?

2. За домовленістю, які ідентифікатори призначаються системним обліковим записам, а які звичайним користувачам?

3. Як дізнатися, чи обліковий запис користувача, який раніше мав доступ до системи, тепер заблоковано? Припустимо, що ваша система використовує тіньові паролі.

Дослідницькі вправи

1. Створіть обліковий запис користувача під назвою `christian` за допомогою команди `useradd -m` і визначте його ідентифікатор користувача (UID), ідентифікатор групи (GID) і оболонку.

2. Визначте назву первинної групи `christian`. Який висновок ви можете зробити?

3. Використовуючи команду `getent`, перегляньте інформацію про старіння пароля для облікового запису користувача `christian`.

4. Додайте групу `editor` до вторинних груп `christian`. Припустимо, що ця група вже містить `emma`, `dave` і `frank` як звичайні члени. Як перевірити, що для цієї групи немає адміністраторів?

5. Виконайте команду `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` та опишіть результат, який вона надає вам у термінах прав доступу до файлів. Які з цих чотирьох файлів затінені з міркувань безпеки? Припустімо, що ваша система використовує тіньові паролі.

Підсумки

На цьому уроці ми дізналися:

- Розташування файлів, які зберігають інформацію про користувачів і групи.
- Керування інформацією про користувачів і групи, що зберігається в базах даних паролів і груп.
- Отримання інформації з баз даних паролів і груп.

У цьому уроці обговорювалися такі файли та команди:

`/etc/passwd`
: Файл, що містить основну інформацію про користувачів.

`/etc/group`
: Файл, що містить основну інформацію про групи.

`/etc/shadow`
: Файл із зашифрованими паролями користувачів.

`/etc/gshadow`
: Файл, що містить зашифровані паролі груп.

`getent`
: Фільтрування бази даних паролів і груп.

Відповіді до вправ посібника

1. Подивіться на наступні вихідні дані і дайте відповідь на такі запитання:

```
# cat /etc/passwd | grep '\(root\|mail\|catherine\|kevin\)'
root:x:0:0:root:/root:/bin/bash
mail:x:8:8:mail:/var/spool/mail:/sbin/nologin
catherine:x:1030:1025:User Chaterine:/home/catherine:/bin/bash
kevin:x:1040:1015:User Kevin:/home/kevin:/bin/bash
# cat /etc/group | grep '\(root\|mail\|db-admin\|app-developer\)'
root:x:0:
mail:x:8:
db-admin:x:1015:emma,grace
app-developer:x:1016:catherine,dave,christian
# cat /etc/shadow | grep '\(root\|mail\|catherine\|kevin\)'
root:$6$1u36Ipok$ljt8ooPMLewAhkQPf.lYgGopAB.jClTO6ljsdczxvkLPkpi/amgp.zyfAN680zrLLp2avvpd
KA0llpssdfcPppOp:18015:0:99999:7:::
mail:*:18015:0:99999:7:::
catherine:$6$ABCD25jlld14hpPthEFGnnssEWw1234yioMpliABCdef1f3478kAfhhAfgbAMjY1/BAeeAsl/FeE
dddKd12345g6kPACcik:18015:20:90:5:::
kevin:$6$DEFGabc123WrLp223fsvp0ddx3dbA7pPPc4LMaa123u6Lp02Lpvm123456pyphhh5ps012vbArL245.P
R1345kkA3Gas12P:18015:0:60:7:2::
# cat /etc/gshadow | grep '\(root\|mail\|db-admin\|app-developer\)'
root:*::
mail:*::
db-admin:!:emma:emma,grace
app-developer:!::catherine,dave,christian
```

   - Що таке ідентифікатор користувача (UID) та ідентифікатор групи (GID) для `root` і `catherine`?

     UID і GID `root` дорівнюють 0 і 0, а UID і GID `catherine` — 1030 і 1025.

   - Як називається первинна група `kevin`? Чи є інші члени цієї групи?

     Назва групи: `db-admin`. У цю групу також входять `emma` і `grace`.

   - Яка оболонка встановлена для `mail`? Що це означає?

     `mail` — це обліковий запис системного користувача, а його оболонка — `/sbin/nologin`. Насправді облікові записи системних користувачів, такі як `mail`, `ftp`, `news` і `daemon`, використовуються для виконання адміністративних завдань, тому для

цих облікових записів слід заборонити звичайний вхід. Ось чому оболонка зазвичай має значення `/sbin/nologin` або `/bin/false`.

- Хто входить до групи `app-developer`? Хто з них є адміністраторами групи, а хто звичайними учасниками?

  Учасниками є `catherine`, `dave` і `christian`, і всі вони є звичайними членами.

- Який мінімальний термін дії пароля для `catherine`? І який максимальний термін служби пароля?

  Мінімальний термін дії пароля становить 20 днів, а максимальний – 90 днів.

- Який період неактивності пароля для `kevin`?

  Термін неактивності пароля - 2 дні. Протягом цього періоду `kevin` повинен оновити пароль, інакше обліковий запис буде вимкнено.

2. За домовленістю, які ідентифікатори призначаються системним обліковим записам, а які звичайним користувачам?

   Системні облікові записи зазвичай мають UID менше 100 або між 500 і 1000, тоді як звичайні користувачі мають UID, починаючи з 1000, хоча деякі застарілі системи можуть починати нумерацію з 500. `root` має UID 0. Пам'ятайте, що `UID_MIN` і `UID_MAX` значення `/etc/login.defs` визначають діапазон UID, які використовуються для створення звичайних користувачів. З точки зору LPI Linux Essentials і LPIC-1, системні облікові записи мають UID менше 1000, а звичайні користувачі мають UID більше 1000.

3. Як дізнатися, чи обліковий запис користувача, який раніше мав доступ до системи, тепер заблоковано? Припустимо, що ваша система використовує тіньові паролі.

   Коли використовуються тіньові паролі, друге поле в `/etc/passwd` містить символ `x` для кожного облікового запису користувача, оскільки зашифровані паролі користувачів зберігаються в `/etc/shadow`. Зокрема, зашифрований пароль облікового запису користувача зберігається у другому полі цього файлу, і якщо він починається зі знаку оклику, обліковий запис заблоковано.

Відповіді до дослідницьких вправ

1. Створіть обліковий запис користувача під назвою `christian` за допомогою команди `useradd -m` і визначте його ідентифікатор користувача (UID), ідентифікатор групи (GID) і оболонку.

```
# useradd -m christian
# cat /etc/passwd | grep christian
christian:x:1050:1060::/home/christian:/bin/bash
```

   UID та GID `christian` — 1050 і 1060 відповідно (третє і четверте поля в `/etc/passwd`). `/bin/bash` — це набір оболонки для цього облікового запису користувача (сьоме поле в `/etc/passwd`).

2. Визначте назву первинної групи `christian`. Який висновок ви можете зробити?

```
# cat /etc/group | grep 1060
christian:x:1060:
```

   Назва первинної групи `christian` — `christian` (перше поле в `/etc/group`). Таким чином, для `USERGROUPS_ENAB` у `/etc/login.defs` встановлено значення yes, щоб `useradd` за замовчуванням створював групу з тим самим іменем, що має обліковий запис користувача.

3. Використовуючи команду `getent`, перегляньте інформацію про старіння пароля для облікового запису користувача `christian`.

```
# getent shadow christian
christian:!:18015:0:99999:7:::
```

   Для облікового запису користувача `christian` не встановлено пароль, і він тепер заблокований (друге поле в `/etc/shadow` містить знак оклику). Для цього облікового запису користувача немає мінімального та максимального терміну дії пароля (четверте та п'яте поля в `/etc/shadow` встановлені на 0 і 99999 днів), тоді як період попередження про пароль встановлено на 7 днів (шосте поле в `/etc/shadow`). Нарешті, немає періоду неактивності (сьоме поле в `/etc/shadow`) і термін дії облікового запису ніколи не закінчується (восьме поле в `/etc/shadow`).

4. Додайте групу `editor` до вторинних груп `christian`. Припустимо, що ця група вже

містить `emma`, `dave` і `frank` як звичайні члени. Як перевірити, що для цієї групи немає адміністраторів?

```
# cat /etc/group | grep editor
editor:x:1100:emma,dave,frank
# usermod -a -G editor christian
# cat /etc/group | grep editor
editor:x:1100:emma,dave,frank,christian
# cat /etc/gshadow | grep editor
editor:!::emma,dave,frank,christian
```

Третє та четверте поля в `/etc/ghadow` містять адміністраторів і звичайних учасників зазначеної групи. Тому, оскільки третє поле для `editor` порожнє, для цієї групи немає адміністраторів (`emma`, `dave`, `frank` і `christian` є звичайними учасниками).

5. Виконайте команду `ls -l /etc/passwd /etc/group /etc/shadow /etc/gshadow` та опишіть результат, який вона надає вам у термінах прав доступу до файлів. Які з цих чотирьох файлів затінені з міркувань безпеки? Припустимо, що ваша система використовує тіньові паролі.

```
# 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
```

Файли `/etc/passwd` і `/etc/group` доступні для читання всіма користувачами та затінені з міркувань безпеки. Коли використовуються тіньові паролі, ви можете побачити `x` у другому полі цих файлів, оскільки зашифровані паролі для користувачів і груп зберігаються в `/etc/shadow` та `/etc/gshadow`, які доступні лише для читання. коренем і, у моїй системі, навіть членами, що належать до групи `shadow`.

107.2 Автоматизація завдань системного адміністрування, планування завдань

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 107.2

Ваговий коефіцієнт

4

Ключові галузі знань

- Керування завданнями за допомогою cron і at.
- Налаштування доступу користувача до служб cron і at.
- Розуміння одиниць таймеру systemd.

Частковий список файлів, термінів та утиліт, що використовуються

- `/etc/cron.{d,daily,hourly,monthly,weekly}/`
- `/etc/at.deny`
- `/etc/at.allow`
- `/etc/crontab`
- `/etc/cron.allow`
- `/etc/cron.deny`
- `/var/spool/cron/`
- `crontab`
- `at`
- `atq`

- `atrm`
- `systemctl`
- `systemd-run`

107.2 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Роздл:	107 Адміністративні завдання
Тема:	107.2 Автоматизація завдань системного адміністрування, планування завдань
Урок:	1 з 2

Вступ

Одним із найважливіших завдань фахового системного адміністратора є планування завдань, які потрібно виконувати на регулярній основі. Наприклад, адміністратор може створювати та автоматизувати завдання для резервного копіювання, оновлення системи та виконання багатьох інших дій, що повторюються. Щоб зробити це, ви можете використати засіб `cron`, який корисний для автоматизації планування періодичних завдань.

Планування завдання за допомогою Cron

У Linux `cron` — це демон, який працює безперервно та прокидається щохвилини, щоб перевірити набір таблиць і знайти завдання для виконання. Ці таблиці відомі як *crontabs* і містять так звані *завдання cron*. Cron підходить для серверів і систем, які постійно увімкнені, тому що кожне завдання cron виконується, лише якщо система працює в запланований час. Його можуть використовувати звичайні користувачі, кожен з яких має власний `crontab`, а також користувач root, який керує системними crontab.

NOTE

У Linux також є функція `anacron`, яка підходить для систем, які можна вимкнути (наприклад, настільних комп'ютерів або ноутбуків). Його може використовувати лише root. Якщо машину вимкнено, коли потрібно виконати завдання `anacron`, вони запустяться під час наступного увімкнення машини. `anacron` не охоплюється сертифікацією LPIC-1.

Користувацькі Crontabs

Користувацькі crontabs — це текстові файли, які керують плануванням визначених користувачем завдань cron. Вони завжди називаються за обліковим записом користувача, який їх створив, але розташування цих файлів залежить від дистрибутива, що використовується (зазвичай це підкаталог `/var/spool/cron`).

Кожен рядок crontab користувача містить шість полів, розділених пробілом:

- Хвилина години (0-59).
- Година доби (0-23).
- День місяця (1-31).
- Місяць року (1-12).
- День тижня (0-7 з неділею=0 або неділею=7).
- Команда для запуску.

Для місяця року та дня тижня можна використовувати перші три літери назви замість відповідної цифри.

Перші п'ять полів вказують, коли потрібно виконати команду, указану в шостому полі, і вони можуть містити одне або кілька значень. Зокрема, ви можете вказати кілька значень за допомогою:

`*` (зірочка)

Посилається на будь-яке значення.

`,` (кома)

Визначає список можливих значень.

`-` (дефіс)

Визначає діапазон можливих значень.

/ (коса риска)

Визначає покрокові значення.

Багато дистрибутивів містять файл `/etc/crontab`, який можна використовувати як довідник для шаблону файлу `cron`. Ось приклад файлу `/etc/crontab` з дистрибутиву Debian:

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
```

Системні Crontabs

Системні crontabs — це текстові файли, які керують плануванням системних завдань cron і можуть редагуватися лише користувачем root. `/etc/crontab` і всі файли в каталозі `/etc/cron.d` є системними crontab.

Більшість дистрибутивів також включають каталоги `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly` та `/etc/cron.monthly`, які містять сценарії для запуску з відповідною частотою. Наприклад, якщо ви хочете запускати сценарій щодня, ви можете розмістити його в `/etc/cron.daily`.

WARNING

Деякі дистрибутиви використовують `/etc/cron.d/hourly`, `/etc/cron.d/daily`, `/etc/cron.d/weekly` та `/etc/cron.d/monthly`. Завжди не забувайте перевіряти правильні каталоги, у яких розміщуєте сценарії, які ви хочете запускати за допомогою cron.

Синтаксис системних crontabs подібний до синтаксису користувацьких crontabs, однак також вимагає додаткового обов'язкового поля, яке вказує, який користувач виконуватиме завдання cron. Таким чином, кожен рядок системного crontab містить сім полів, розділених пробілом:

- Хвилина години (0-59).

- Година доби (0-23).
- День місяця (1-31).
- Місяць року (1-12).
- День тижня (0-7 з неділею=0 або неділею=7).
- Ім'я облікового запису користувача, який буде використовуватися під час виконання команди.
- Команда для запуску.

Що стосується crontab користувача, ви можете вказати кілька значень для полів часу за допомогою операторів `*`, `,` , `-` і `/`. Ви також можете вказати місяць року та день тижня першими трьома літерами назви замість відповідної цифри.

Специфікації конкретного часу

Під час редагування файлів crontab ви також можете використовувати спеціальні ярлики в перших п'яти стовпцях замість визначення конкретного часу:

`@reboot`
: Запускати вказане завдання один раз після перезавантаження.

`@hourly`
: Запускати вказане завдання раз на годину на початку години.

`@daily` (або `@midnight`)
: Запускати вказане завдання раз на день опівночі.

`@weekly`
: Запускати вказане завдання раз на тиждень опівночі в неділю.

`@monthly`
: Запускати вказане завдання раз на місяць опівночі першого дня місяця.

`@yearly` (або `@annually`)
: Запускати вказане завдання один раз на рік опівночі 1 січня.

Змінні Crontab

У файлі crontab іноді є призначення змінних, які визначаються перед оголошенням запланованих завдань. Зазвичай встановлюються такі змінні середовища:

`HOME`

Каталог, де `cron` викликає команди (за замовчуванням домашній каталог користувача).

`MAILTO`

Ім'я користувача або адреса, на яку надсилається стандартний вивід і помилка (за замовчуванням власник crontab). Кілька значень, розділених комами, також дозволені, а порожнє значення вказує на те, що пошту не слід надсилати.

`PATH`

Шлях, де можна знайти команди.

`SHELL`

Оболонка, що використовується (за умовчанням `/bin/sh`).

Створення завдань користувача Cron

Команда `crontab` використовується для підтримки файлів crontab для окремих користувачів. Зокрема, ви можете запустити команду `crontab -e`, щоб відредагувати свій власний файл crontab або створити його, якщо він ще не існує.

```
$ crontab -e
no crontab for frank - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano        < ---- easiest
  3. /usr/bin/emacs24
  4. /usr/bin/vim.tiny

Choose 1-4 [2]:
```

За замовчуванням команда `crontab` відкриває редактор, визначений змінними середовища `VISUAL` або `EDITOR`, щоб ви могли почати редагувати свій файл crontab за допомогою бажаного редактора. Деякі дистрибутиви, як показано в прикладі вище, дозволяють вибрати редактор зі списку, коли `crontab` запускається вперше.

Якщо ви хочете запускати сценарій `foo.sh`, який знаходиться у вашому домашньому каталозі щодня о 10:00 ранку, ви можете додати такий рядок до свого файлу crontab:

```
0 10 * * * /home/frank/foo.sh
```

Розглянемо наступні приклади записів crontab:

```
0,15,30,45 08 * * 2 /home/frank/bar.sh
30 20 1-15 1,6 1-5 /home/frank/foobar.sh
```

У першому рядку скрипт `bar.sh` виконується щовівторка о 08:00, о 08:15, о 08:30 та о 08:45. У другому рядку сценарій `foobar.sh` виконується о 20:30 з понеділка по п'ятницю протягом перших п'ятнадцяти днів січня та червня.

WARNING

Хоча файли crontab можна редагувати вручну, завжди рекомендується використовувати команду `crontab`. Права доступу до файлів crontab зазвичай дозволяють лише редагувати їх за допомогою команди `crontab`.

На додаток до параметра `-e`, згаданого вище, команда `crontab` має інші корисні параметри:

`-l`

Відображає поточний crontab у стандартному виведенні.

`-r`

Видаляє поточний crontab.

`-u`

Визначає ім'я користувача, crontab якого потрібно змінити. Цей параметр потребує привілеїв root і дозволяє користувачеві root редагувати файли користувача crontab.

Створення системних завдань Cron

На відміну від користувацьких crontab, системні crontab оновлюються за допомогою редактора: тому вам не потрібно запускати команду `crontab`, щоб редагувати `/etc/crontab` та файли в `/etc/cron.d`. Пам'ятайте, що під час редагування системних crontabs ви повинні вказати обліковий запис, який використовуватиметься для запуску завдання cron (зазвичай користувач root).

Наприклад, якщо ви хочете запускати сценарій `barfoo.sh`, розташований у каталозі `/root` щодня о 01:30 ночі, ви можете відкрити `/etc/crontab` за допомогою редактора, якому ви віддаєте перевагу, і додати такий рядок:

```
30 01 * * * root /root/barfoo.sh >>/root/output.log 2>>/root/error.log
```

У наведеному вище прикладі вихідні дані завдання додаються до `/root/output.log`, тоді як помилки додаються до `/root/error.log`.

WARNING

Якщо вихідні дані не перенаправляються у файл, як у наведеному вище прикладі (або для змінної `MAILTO` встановлено порожнє значення), увесь вивід із завдання cron буде надсилатись користувачеві електронною поштою. Поширеною практикою є перенаправлення стандартного виводу до `/dev/null` (або до файлу для подальшого перегляду, якщо необхідно) і не перенаправлення стандартної помилки. Таким чином користувач буде негайно повідомлений електронною поштою про будь-які помилки.

Налаштування доступу до планування завдань

У Linux файли `/etc/cron.allow` і `/etc/cron.deny` використовуються для встановлення обмежень `crontab`. Зокрема, вони використовуються для дозволу або заборони планування завдань cron для різних користувачів. Якщо `/etc/cron.allow` існує, лише некореневі користувачі, перелічені в ньому, можуть планувати завдання cron за допомогою команди `crontab`. Якщо `/etc/cron.allow` не існує, але `/etc/cron.deny` існує, лише не-root користувачі, перелічені в цьому файлі, не можуть планувати завдання cron за допомогою команди `crontab` (у цьому випадку порожній `/ etc/cron.deny` означає, що кожному користувачеві дозволено планувати завдання cron за допомогою `crontab`). Якщо жоден із цих файлів не існує, доступ користувача до планування завдань cron залежить від дистрибутива, що використовується.

NOTE

Файли `/etc/cron.allow` і `/etc/cron.deny` містять список імен користувачів, кожне в окремому рядку.

Альтернатива Cron

Використовуючи systemd як менеджер системи та послуг, ви можете встановити *timers* як альтернативу `cron` для планування своїх завдань. Таймери — це системні файли одиниць, ідентифіковані суфіксом `.timer`, і для кожного з них повинен існувати відповідний unit-файл, який описує unit, який буде активований після закінчення таймера. За замовчуванням `timer` активує службу з такою ж назвою, за винятком суфікса.

Таймер містить розділ `[Timer]`, який визначає час виконання запланованих завдань. Зокрема, ви можете використовувати опцію `OnCalendar=`, щоб визначити *таймери реального часу*, які працюють так само, як і завдання cron (вони базуються на виразах подій календаря). Параметр `OnCalendar=` вимагає такого синтаксису:

```
DayOfWeek Year-Month-Day Hour:Minute:Second
```

Параметр `DayOfWeek` є необов'язковим. Оператори `*`, `/` і `,` мають те саме значення, що й ті, що використовуються для завдань cron, тоді як ви можете використовувати `..` між двома значеннями, щоб позначити безперервний діапазон. Для специфікації `DayOfWeek` ви можете використовувати перші три літери назви або повну назву.

NOTE Ви також можете визначити *монотонні таймери*, які активуються через деякий час, що минув із певної початкової точки (наприклад, коли машина завантажилася або коли сам таймер активовано).

Наприклад, якщо ви хочете запустити службу під назвою `/etc/systemd/system/foobar.service` о 05:30 першого понеділка кожного місяця, ви можете додати наступні рядки у відповідний unit-файл `/etc/systemd/system/foobar.timer`.

```
[Unit]
Description=Run the foobar service

[Timer]
OnCalendar=Mon *-*-1..7 05:30:00
Persistent=true

[Install]
WantedBy=timers.target
```

Створивши новий таймер, ви можете ввімкнути його та запустити, виконавши наступні команди від імені root:

```
# systemctl enable foobar.timer
# systemctl start foobar.timer
```

Ви можете змінити частоту запланованого завдання, змінивши значення `OnCalendar`, а потім ввівши команду `systemctl daemon-reload`.

Нарешті, якщо ви бажаєте переглянути список активних таймерів, упорядкованих за часом закінчення, ви можете скористатися командою `systemctl list-timers`. Ви можете додати опцію `--all`, щоб також бачити неактивні таймери.

NOTE Пам'ятайте, що таймери реєструються в журналі systemd, і ви можете

> переглядати журнали різних модулів за допомогою команди `journalctl`. Також пам'ятайте, що якщо ви дієте як звичайний користувач, вам потрібно використовувати опцію `--user` команд `systemctl` і `journalctl`.

Замість довшої нормалізованої форми, згаданої вище, ви можете використовувати деякі спеціальні вирази, які описують конкретні частоти для виконання завдання:

`hourly`
: Запускати вказане завдання раз на годину на початку години.

`daily`
: Запускати вказане завдання раз на день опівночі.

`weekly`
: Запускати вказане завдання раз на тиждень опівночі понеділка.

`monthly`
: Запускати вказане завдання раз на місяць опівночі першого дня місяця.

`yearly`
: Запускати вказане завдання раз на рік опівночі першого січня.

Повний перелік специфікацій часу та дати можна переглянути на man-сторінках у `systemd.timer(5)`.

Вправи до посібника

1. Для кожного з наступних ярликів `crontab` вкажіть відповідну специфікацію часу (тобто перші п’ять стовпців у файлі `crontab` користувача):

`@hourly`	
`@daily`	
`@weekly`	
`@monthly`	
`@annually`	

2. Для кожного з наступних ярликів `OnCalendar` вкажіть відповідну специфікацію часу (довша нормалізована форма):

`hourly`	
`daily`	
`weekly`	
`monthly`	
`yearly`	

3. Поясніть значення наступних специфікацій часу, знайдених у файлі `crontab`:

`30 13 * * 1-5`	
`00 09-18 * * *`	
`30 08 1 1 *`	
`0,20,40 11 * * Sun`	
`00 09 10-20 1-3 *`	
`*/20 * * * *`	

4. Поясніть значення наступних специфікацій часу, які використовуються в параметрі `OnCalendar` файлу таймера:

`*-*-* 08:30:00`	
`Sat,Sun *-*-* 05:00:00`	

`*-*-01 13:15,30,45:00`	
`Fri *-09..12-* 16:20:00`	
`Mon,Tue *-*-1,15 08:30:00`	
`*-*-* *:00/05:00`	

Дослідницькі вправи

1. Якщо припустити, що ви маєте право планувати завдання за допомогою `cron` як звичайний користувач, яку команду ви використаєте, щоб створити власний файл crontab?

2. Створіть просте заплановане завдання, яке виконує команду `date` щоп'ятниці о 13:00. Де можна побачити результати цієї роботи?

3. Створіть інше заплановане завдання, яке виконує сценарій `foobar.sh` щохвилини, перенаправляючи вихідні дані у файл `output.log` у вашому домашньому каталозі, щоб вам електронною поштою надсилалися лише стандартні помилки.

4. Подивіться на запис `crontab` щойно створеного запланованого завдання. Чому не потрібно вказувати абсолютний шлях до файлу, у якому зберігається стандартний вивід? І чому ви можете використовувати команду `./foobar.sh` для виконання сценарію?

5. Відредагуйте попередній запис `crontab`, видаливши переспрямування виводу та вимкнувши перше завдання cron, яке ви створили.

6. Як ви можете надіслати результати та помилки вашого запланованого завдання до облікового запису користувача `emma` електронною поштою? І як можна уникнути надсилання стандартного виведення та помилки електронною поштою?

7. Виконайте команду ls -l /usr/bin/crontab. Який спеціальний біт встановлений і яке його значення?

Підсумки

На цьому уроці ви дізналися про:

- Використання `cron` для запуску завдань через регулярні проміжки часу.
- Керування завданнями cron.
- Налаштування доступу користувача до планування завдань cron.
- Зрозуміли роль таймерів systemd як альтернативи `cron`.

У цьому уроці обговорювалися такі команди та файли:

`crontab`

Підтримує файли `crontab` для окремих користувачів.

`/etc/cron.allow` та **`/etc/cron.deny`**

Окремі файли, які використовуються для встановлення обмежень `crontab`.

`/etc/crontab`

Системний файл crontab.

`/etc/cron.d`

Каталог, який містить системні файли crontab.

`systemctl`

Керує системою systemd і диспетчером служб. Стосовно таймерів його можна використовувати для їх увімкнення та запуску.

Відповіді до вправ посібника

1. Для кожного з наступних ярликів `crontab` вкажіть відповідну специфікацію часу (тобто перші п’ять стовпців у файлі `crontab` користувача):

`@hourly`	`0 * * * *`
`@daily`	`0 0 * * *`
`@weekly`	`0 0 * * 0`
`@monthly`	`0 0 1 * *`
`@annually`	`0 0 1 1 *`

2. Для кожного з наступних ярликів `OnCalendar` вкажіть відповідну специфікацію часу (довша нормалізована форма):

`hourly`	`*-*-* *:00:00`
`daily`	`*-*-* 00:00:00`
`weekly`	`Mon *-*-* 00:00:00`
`monthly`	`*-*-01 00:00:00`
`yearly`	`*-01-01 00:00:00`

3. Поясніть значення наступних специфікацій часу, знайдених у файлі `crontab`:

`30 13 * * 1-5`	О 13:30 щодня з понеділка по п’ятницю
`00 09-18 * * *`	Щодня та щогодини з 09:00 до 18:00
`30 08 1 1 *`	О 08:30 ранку першого січня
`0,20,40 11 * * Sun`	Щонеділі об 11:00, 11:20 та 11:40
`00 09 10-20 1-3 *`	О 09:00 з 10 по 20 січня, лютого та березня
`*/20 * * * *`	Кожні двадцять хвилин

4. Поясніть значення наступних специфікацій часу, які використовуються в параметрі `OnCalendar` файлу таймера:

`*-*-* 08:30:00`	Щодня о 08:30 год
`Sat,Sun *-*-* 05:00:00`	У суботу та неділю о 05:00 ранку

`*-*-01 13:15,30,45:00`	О 13:15, 13:30 та 13:45 першого числа місяця
`Fri *-09..12-* 16:20:00`	О 16:20 кожної п’ятниці у вересні, жовтні, листопаді та грудні
`Mon,Tue *-*-1,15 08:30:00`	О 08:30 ранку першого або п’ятнадцятого числа кожного місяця, лише якщо день припадає на понеділок або вівторок
`*-*-* *:00/05:00`	Кожні п’ять хвилин

Відповіді до дослідницьких вправ

1. Якщо припустити, що ви маєте право планувати завдання за допомогою `cron` як звичайний користувач, яку команду ви використаєте, щоб створити власний файл crontab?

```
dave@hostname ~ $ crontab -e
no crontab for dave - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano        < ---- easiest
  3. /usr/bin/emacs24
  4. /usr/bin/vim.tiny

Choose 1-4 [2]:
```

2. Створіть просте заплановане завдання, яке виконує команду `date` щоп'ятниці о 13:00. Де можна побачити результати цієї роботи?

```
00 13 * * 5 date
```

   Вихідні дані надсилаються користувачеві; щоб переглянути їх, ви можете скористатися командою `mail`.

3. Створіть інше заплановане завдання, яке виконує сценарій `foobar.sh` щохвилини, перенаправляючи вихідні дані у файл `output.log` у вашому домашньому каталозі, щоб вам електронною поштою надсилалися лише стандартні помилки.

```
*/1 * * * * ./foobar.sh >> output.log
```

4. Подивіться на запис `crontab` щойно створеного запланованого завдання. Чому не потрібно вказувати абсолютний шлях до файлу, у якому зберігається стандартний вивід? І чому ви можете використовувати команду `./foobar.sh` для виконання сценарію?

   `cron` викликає команди з домашнього каталогу користувача, якщо інше розташування не вказано змінною середовища `HOME` у файлі `crontab`. З цієї причини ви можете використати відносний шлях вихідного файлу та запустити сценарій за допомогою

`./foobar.sh`.

5. Відредагуйте попередній запис `crontab`, видаливши переспрямування виводу та вимкнувши перше завдання cron, яке ви створили.

```
#00 13 * * 5 date
*/1 * * * * ./foobar.sh
```

   Щоб вимкнути завдання cron, ви можете просто закоментувати відповідний рядок у файлі `crontab`.

6. Як ви можете надіслати результати та помилки вашого запланованого завдання до облікового запису користувача `emma` електронною поштою? І як можна уникнути надсилання стандартного виведення та помилки електронною поштою?

   Щоб надіслати стандартний вихід і помилку до `emma`, ви повинні встановити змінну середовища `MAILTO` у вашому файлі `crontab` таким чином:

```
MAILTO="емма"
```

   Щоб повідомити `cron`, що пошта не повинна надсилатися, ви можете призначити порожнє значення змінній середовища `MAILTO`.

```
MAILTO=""
```

7. Виконайте команду ls -l /usr/bin/crontab. Який спеціальний біт встановлений і яке його значення?

```
$ ls -l /usr/bin/crontab
-rwxr-sr-x 1 root crontab 25104 feb 10  2015 /usr/bin/crontab
```

   Команда `crontab` має встановлений біт SGID (символ `s` замість позначки виконуваного файлу для групи), що означає, що вона виконується з привілеями групи (отже, `crontab`). Ось чому звичайні користувачі можуть редагувати свій файл `crontab` за допомогою команди `crontab`. Зауважте, що багато дистрибутивів мають права доступу до файлів, налаштовані таким чином, що файли `crontab` можна редагувати лише за допомогою команди `crontab`.

107.2 Урок 2

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	107 Адміністративні завдання
Тема:	107.2 Автоматизація завдань системного адміністрування, планування завдань
Урок:	2 з 2

Вступ

Як ви дізналися з попереднього уроку, ви можете планувати звичайні завдання за допомогою таймерів cron або systemd, але іноді вам може знадобитися виконати завдання в певний час у майбутньому лише один раз. Для цього ви можете скористатися іншою потужною утилітою: командою `at`.

Планування роботи за допомогою `at`

Команда `at` використовується для одноразового планування завдань і вимагає лише вказати час виконання завдання в майбутньому. Після введення `at` у командному рядку з подальшою специфікацією часу ви побачите підказку `at`, де зможете визначити команди, які потрібно виконати. Ви можете вийти з підказки, натиснувши послідовність клавіш `Ctrl` + `D`.

```
$ at now +5 minutes
warning: commands will be executed using /bin/sh
at> date
```

```
at> Ctrl+D
job 12 at Sat Sep 14 09:15:00 2019
```

Завдання `at` у наведеному вище прикладі просто виконує команду `date` через п'ять хвилин. Подібно до `cron`, стандартне виведення і помилка надсилаються вам електронною поштою. Зауважте, що демон `atd` має бути запущений у системі, щоб ви могли використовувати планування завдань `at`.

NOTE У Linux команда `batch` подібна до `at`, однак завдання `batch` виконуються лише тоді, коли навантаження на систему досить низьке, щоб це дозволити.

Найважливіші параметри, які застосовуються до команди `at`:

`-c`
Вивести команди певного ID завдання у стандартне виведення.

`-d`
Видалити завдання на основі їх ID. Це псевдонім для `atrm`.

`-f`
Читати завдання з файлу замість стандартного введення.

`-l`
Показати незавершені завдання користувача. Якщо користувач root, виводяться всі завдання всіх користувачів. Це псевдонім для `atq`.

`-m`
Надіслати пошту користувачеві після завершення завдання, навіть якщо результату не було.

`-q`
Вказати чергу у вигляді однієї літери від `a` до `z` і від `A` до `Z` (за умовчанням `a` для `at` і `b` для `batch`). Роботи в чергах з найвищими буквами виконуються з підвищеною акуратністю. Завдання, надіслані до черги з великої літери, розглядаються як пакетні завдання.

`-v`
Показати час виконання завдання перед його читанням.

Список запланованих завдань за допомогою `atq`

Тепер давайте заплануємо ще два завдання `at`: перше виконує сценарій `foo.sh` о 09:30, а друге виконує сценарій `bar.sh` через одну годину.

```
$ at 09:30 AM
warning: commands will be executed using /bin/sh
at> ./foo.sh
at> Ctrl+D
job 13 at Sat Sep 14 09:30:00 2019
$ at now +2 hours
warning: commands will be executed using /bin/sh
at> ./bar.sh
at> Ctrl+D
job 14 at Sat Sep 14 11:10:00 2019
```

Щоб переглянути список незавершених завдань, ви можете скористатися командою `atq`, яка показує таку інформацію для кожного завдання: ідентифікатор завдання, дата виконання завдання, час виконання завдання, черга та ім'я користувача.

```
$ atq
14      Sat Sep  14 11:10:00 2019 a frank
13      Sat Sep  14 09:30:00 2019 a frank
12      Sat Sep  14 09:15:00 2019 a frank
```

Пам'ятайте, що команда `at -l` є псевдонімом `atq`.

NOTE Якщо ви запустите `atq` як root, він відобразить завдання в черзі для всіх користувачів.

Видалення завдання за допомогою `atrm`

Якщо ви бажаєте видалити завдання `at`, ви можете використати команду `atrm`, за якою слід вказати ідентифікатор завдання. Наприклад, щоб видалити завдання з ID 14, ви можете виконати наступне:

```
$ atrm 14
```

Ви можете видалити кілька завдань за допомогою `atrm`, вказавши кілька ідентифікаторів, розділених пробілами. Пам'ятайте, що команда `at -d` є псевдонімом `atrm`.

NOTE Якщо ви запускаєте `atrm` як root, ви можете видалити завдання всіх користувачів.

Налаштуваня доступу до планування завдань

Авторизація для звичайних користувачів щодо планування завдань `at` визначається файлами `/etc/at.allow` і `/etc/at.deny`. Якщо `/etc/at.allow` існує, лише некореневі користувачі, перелічені в ньому, можуть планувати завдання `at`. Якщо `/etc/at.allow` не існує, але `/etc/at.deny` існує, лише некореневі користувачі, перелічені в ньому, не можуть планувати завдання `at` (у цьому випадку порожній `/etc/at.deny` файл означає, що кожному користувачеві дозволено планувати завдання `at`). Якщо жоден із цих файлів не існує, доступ користувача до планування завдань `at` залежить від дистрибутива, що використовується.

Специфікації часу

Ви можете вказати, коли виконувати певне завдання `at`, використовуючи формат `HH:MM`, який необов'язково супроводжується AM або PM у випадку 12-годинного формату. Якщо зазначений час уже минув, передбачається наступний день. Якщо ви хочете запланувати конкретну дату, на яку буде виконано завдання, ви повинні додати інформацію про дату після часу, використовуючи одну з таких форм: `month-name day-of-month`, `month-name day-of-month year`, `MMDDYY`, `MM/DD/YY`, `DD.MM.YY` та `YYYY-MM-DD`).

Також приймаються такі ключові слова: `midnight`, `noon`, `teatime` (4 pm) і `now`, за якими йде знак плюс (+) і період часу (хвилини, години, дні та тижні). Нарешті, ви можете вказати `at` виконати завдання сьогодні або завтра, додавши до часу суфікс слова `today` або `tomorrow`. Наприклад, ви можете використовувати `at 07:15 AM Jan 01`, щоб виконати завдання о 07:15 ранку 1 січня, і `at now +5 minutes`, щоб виконати завдання через п'ять хвилин. Ви можете прочитати файл `timespec` у дереві `/usr/share`, щоб дізнатися більше про точне визначення специфікацій часу.

Альтернатива `at`

Використовуючи systemd як менеджер системи та служб, ви також можете планувати одноразові завдання за допомогою команди `systemd-run`. Зазвичай вона використовується для створення блоку тимчасового таймера, щоб команда виконувалася в певний час без необхідності створення службового файлу. Наприклад, діючи від імені root, ви можете виконати команду `date` об 11:30 ранку 2019/10/06, використовуючи наступне:

```
# systemd-run --on-calendar='2019-10-06 11:30' date
```

Якщо ви хочете запустити сценарій `foo.sh`, розташований у вашому поточному робочому каталозі, через дві хвилини ви можете використати:

```
# systemd-run --on-active="2m" ./foo.sh
```

Зверніться до man-сторінок, щоб дізнатися про всі можливі способи використання `systemd-run` із `systemd-run(1)`.

NOTE

Пам'ятайте, що таймери реєструються в журналі systemd, і ви можете переглядати журнали різних модулів за допомогою команди `journalctl`. Також пам'ятайте, що якщо ви дієте як звичайний користувач, вам потрібно використовувати опцію `--user` команд `systemd-run` і `journalctl`.

Вправи до посібника

1. Для кожної з наведених нижче специфікацій часу вкажіть, яка дійсна, а яка недійсна для `at`:

`at 08:30 AM next week`	
`at midday`	
`at 01-01-2020 07:30 PM`	
`at 21:50 01.01.20`	
`at now +4 days`	
`at 10:15 PM 31/03/2021`	
`at tomorrow 08:30 AM`	

2. Якщо ви запланували завдання за допомогою `at`, як ви можете їх переглянути?

3. Які команди можна використовувати для перегляду незавершених завдань? Якими командами ви б їх видалили?

4. Яка команда з systemd використовується як альтернатива `at`?

Дослідницькі вправи

1. Створіть завдання `at`, яке запускає сценарій `foo.sh`, розташований у вашому домашньому каталозі, о 10:30 31 жовтня. Припустимо, ви дієте як звичайний користувач.

2. Увійдіть у систему як інший звичайний користувач і завтра о 10:00 створіть інше завдання `at`, яке завтра о 10:00 запускає сценарій `bar.sh`. Припустимо, що сценарій знаходиться в домашньому каталозі користувача.

3. Увійдіть у систему як інший звичайний користувач і створіть інше завдання `at`, яке запускає сценарій `foobar.sh` лише через 30 хвилин. Припустимо, що сценарій знаходиться в домашньому каталозі користувача.

4. Тепер від імені користувача root виконайте команду `atq`, щоб переглянути заплановані завдання `at` для всіх користувачів. Що станеться, якщо звичайний користувач виконає цю команду?

5. Як root видаліть усі ці незавершені завдання `at` за допомогою однієї команди.

6. Виконайте команду `ls -l /usr/bin/at` і перевірте її дозволи.

Підсумки

В цьому уроці ми вивчили:

- Використання `at` для запуску одноразових завдань в певний час.
- Керування завданнями `at`.
- Налаштування доступу користувача до планування завдань `at`.
- Використання `systemd-run` в якості альтернативи `at`.

У цьому уроці обговорювалися такі файли та команди:

`at`
: Виконує команди в заданий час.

`atq`
: Виводить перелік незавершених завдань користувача, якщо користувач не є суперкористувачем.

`atrm`
: Видаляє завдання `at`, ідентифіковані їх номерами.

`/etc/at.allow` and `/etc/at.deny`
: Окремі файли, які використовуються для встановлення обмежень `at`.

`systemd-run`
: Створює і запускає перехідний блок `timer` як альтернативу `at` для одноразового планування.

Відповіді до вправ посібника

1. Для кожної з наведених нижче специфікацій часу вкажіть, яка дійсна, а яка недійсна для `at`:

`at 08:30 AM next week`	Дійсна
`at midday`	Недійсна
`at 01-01-2020 07:30 PM`	Недійсна
`at 21:50 01.01.20`	Дійсна
`at now +4 days`	Дійсна
`at 10:15 PM 31/03/2021`	Недійсна
`at tomorrow 08:30 AM monotonic`	Недійсна

2. Якщо ви запланували завдання за допомогою `at`, як ви можете їх переглянути?

   Ви можете використовувати команду `at -c`, за якою слідує ідентифікатор завдання, команди якого ви хочете переглянути. Зауважте, що вихідні дані також містять більшу частину середовища, яке було активним на момент запланованого завдання. Пам'ятайте, що root може переглядати завдання всіх користувачів.

3. Які команди можна використовувати для перегляду незавершених завдань? Якими командами ви б їх видалили?

   Ви можете використати команду `at -l`, щоб переглянути незавершені завдання, а також команду `at -d`, щоб видалити свої завдання. `at -l` є псевдонімом для `atq`, а `at -d` є псевдонімом для `atrm`. Пам'ятайте, що root може дивитися та видаляти завдання всіх користувачів.

4. Яка команда з systemd використовується як альтернатива `at`?

   Команду `systemd-run` можна використовувати як альтернативу `at` для планування одноразових завдань. Наприклад, ви можете використовувати її для запуску команд у певний час, визначаючи *календарний таймер* або *монотонний таймер* відносно різних початкових точок.

Відповіді до дослідницьких вправ

1. Створіть завдання `at`, яка запускає сценарій `foo.sh`, розташований у вашому домашньому каталозі, о 10:30 31 жовтня. Припустимо, ви дієте як звичайний користувач.

```
$ at 10:30 AM October 31
warning: commands will be executed using /bin/sh
at> ./foo.sh
at> Ctrl+D
job 50 at Thu Oct 31 10:30:00 2019
```

2. Увійдіть у систему як інший звичайний користувачі створіть інше завдання `at`, яке завтра о 10:00 запускає сценарій `bar.sh`. Припустимо, що сценарій знаходиться в домашньому каталозі користувача.

```
$ at 10:00 AM tomorrow
warning: commands will be executed using /bin/sh
at> ./bar.sh
at> Ctrl+D
job 51 at Sun Oct 6 10:00:00 2019
```

3. Увійдіть у систему як інший звичайний користувач і створіть інше завдання `at`, яке завтра о 10:00 запускає сценарій `bar.sh`. Припустимо, що сценарій знаходиться в домашньому каталозі користувача.

```
$ at now +30 minutes
warning: commands will be executed using /bin/sh
at> ./foobar.sh
at> Ctrl+D
job 52 at Sat Oct 5 10:19:00 2019
```

4. Тепер від імені користувача root виконайте команду `atq`, щоб переглянути заплановані завдання `at` для всіх користувачів. Що станеться, якщо звичайний користувач виконає цю команду?

```
# atq
52      Sat Oct  5 10:19:00 2019 a dave
50      Thu Oct 31 10:30:00 2019 a frank
```

```
51      Sun Oct  6 10:00:00 2019 a emma
```

Якщо ви виконуєте команду `atq` від імені користувача root, буде показано всі незавершені завдання `at` для всіх користувачів. Якщо ви запускаєте команду як звичайний користувач, у списку будуть лише ваші власні незавершені завдання.

5. Як root видаліть усі ці незавершені завдання `at` за допомогою однієї команди.

```
# atrm 50 51 52
```

6. Від імені користувача root виконайте команду ls -l /usr/bin/at і перевірте її дозволи.

```
# ls -l /usr/bin/at
-rwsr-sr-x 1 daemon daemon 43762 Dec  1  2015 /usr/bin/at
```

У цьому дистрибутиві команда `at` має встановлені біти SUID (символ `s` замість прапора виконуваного файлу для власника) і SGID (символ `s` замість прапора виконуваного файлу для групи), це означає що команда виконується з привілеями власника та групи файлу (`daemon` для обох). Ось чому звичайні користувачі можуть планувати завдання за допомогою `at`.

107.3 Локалізація та інтернаціоналізація

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 107.3

Ваговий коефіцієнт

3

Ключові галузі знань

- Налаштування параметрів мови та змінних середовища.
- Налаштування параметрів часового поясу та змінних середовища.

Частковий список файлів, термінів та утиліт, що використовуються

- `/etc/timezone`
- `/etc/localtime`
- `/usr/share/zoneinfo/`
- `LC_*`
- `LC_ALL`
- `LANG`
- `TZ`
- `/usr/bin/locale`
- `tzselect`
- `timedatectl`
- `date`
- `iconv`

- `UTF-8`
- `ISO-8859`
- ASCII
- Unicode

107.3 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	107 Адміністративні завдання
Тема:	107.3 Локалізація та інтернаціоналізація
Урок:	1 з 1

Вступ

Усі основні дистрибутиви Linux можуть бути налаштовані на використання користувацьких налаштувань локалізації. Ці налаштування включають визначення, пов'язані з регіоном і мовою, наприклад часовий пояс, мову інтерфейсу, а також кодування символів, і їх можна змінити під час встановлення операційної системи або будь-коли після цього.

Програми покладаються на змінні середовища, файли конфігурації системи та команди, щоб визначити правильний час і мову для використання; тому більшість дистрибутивів Linux мають стандартизований спосіб налаштування часу та параметрів локалізації. Ці налаштування важливі не лише для покращення взаємодії з користувачем, але й для забезпечення правильного розрахунку часу системних подій, важливих, наприклад, для звітування про проблеми, пов'язані з безпекою.

Щоб мати можливість представити будь-який письмовий текст, незалежно від усної мови, сучасні операційні системи потребують еталонного *стандарту кодування символів*, і системи Linux не є виключенням. Оскільки комп'ютери можуть працювати лише з числами, текстовий символ – це не що інше, як число, пов'язане з графічним символом.

Різні комп'ютерні платформи можуть пов'язувати різні числові значення з тим самим символом, тому для їх сумісності необхідний загальний стандарт кодування символів. Текстовий документ, створений в одній системі, можна буде прочитати в іншій системі, лише якщо обидва домовляться про формат кодування та про те, яке число пов'язане з яким символом, або принаймні якщо вони знають, як проводити конвертацію між двома стандартами.

Гетерогенний характер налаштувань локалізації в системах на базі Linux призводить до тонких відмінностей між дистрибутивами. Незважаючи на ці відмінності, усі дистрибутиви мають однакові основні інструменти та концепції для налаштування аспектів інтернаціоналізації системи.

Часові пояси

Часові пояси — це приблизно пропорційні дискретні смуги земної поверхні, що охоплюють еквівалент однієї години, тобто регіони світу, які мають однакову годину доби в будь-який момент часу. Оскільки немає жодної довготи, яку можна вважати початком дня для всього світу, часові пояси відносяться до *початкового меридіана*, де кут довготи Землі визначається як 0. Час на початковому меридіані називається *Всесвітній координований час* (Coordinated Universal Time), за домовленістю скорочено до UTC. З практичних причин часові пояси не відповідають точній поздовжній відстані від точки відліку (нульового меридіана). Натомість часові пояси штучно пристосовуються до кордонів країн або інших значущих поділів.

Політичні поділи настільки актуальні, що часові пояси називаються на честь якогось головного географічного агента в цій конкретній території, зазвичай на основі назви великої країни чи міста в зоні. Тим не менш, часові пояси поділяються відповідно до їхнього часового зміщення відносно UTC, і це зміщення також може використовуватися для позначення зони, про яку йде мова. Часовий пояс *GMT-5*, наприклад, вказує на регіон, для якого час UTC на п'ять годин випереджає, тобто цей регіон відстає від UTC на 5 годин. Так само часовий пояс *GMT+3* вказує на регіон, для якого час UTC відстає на три години. Термін GMT — від *Greenwich Mean Time* (час за Грінвичем) — використовується як синонім UTC у назвах зон.

До під'єднаного комп'ютера можна отримати доступ із різних куточків світу, тому рекомендовано встановити апаратний годинник на UTC (часовий пояс GMT+0) і залишити вибір часового поясу для кожного окремого випадку. Хмарні служби, наприклад, зазвичай налаштовані на використання UTC, оскільки це може допомогти зменшити випадкові невідповідності між місцевим часом і часом на клієнтах або на інших серверах. Навпаки, користувачі, які відкривають віддалений сеанс на сервері, можуть захотіти використовувати свій місцевий часовий пояс. Таким чином, операційна система повинна

встановити правильний часовий пояс відповідно до кожного випадку.

Окрім поточної дати та часу, команда `date` також виведе поточний налаштований часовий пояс:

```
$ date
Mon Oct 21 10:45:21 -03 2019
```

Зсув відносно UTC задається значенням `-03`. Це означає, що відображений час на три години менший за UTC. Таким чином, час UTC на три години вперед, що робить *GMT-3* відповідним часовим поясом для даного часу. Команда `timedatectl, яка доступна в дистрибутивах, що використовують systemd, показує більше деталей про системний час і дату:

```
$ timedatectl
                      Local time: Sat 2019-10-19 17:53:18 -03
                  Universal time: Sat 2019-10-19 20:53:18 UTC
                        RTC time: Sat 2019-10-19 20:53:18
                       Time zone: America/Sao_Paulo (-03, -0300)
       System clock synchronized: yes
systemd-timesyncd.service active: yes
                 RTC in local TZ: no
```

Як видно в рядку `Time zone`, також приймаються назви часових поясів на основі місцевостей, наприклад, `America/Sao_Paulo`. Часовий пояс за замовчуванням для системи зберігається у файлі `/etc/timezone`, або за повною описовою назвою зони, або за зміщенням. Загальні назви часових поясів, надані зсувом відносно UTC, мають містити `Etc` як першу частину назви. Отже, щоб встановити часовий пояс за замовчуванням на GMT+3, назва часового поясу має бути `Etc/GMT+3`:

```
$ cat /etc/timezone
Etc/GMT+3
```

Хоча назви часових поясів, засновані на місцевостях, не вимагають зміщення часу для роботи, їх не так просто вибрати. Одна зона може мати більше однієї назви, що ускладнює її запам'ятовування. Щоб полегшити цю проблему, команда `tzselect` пропонує інтерактивний метод, який допоможе користувачеві правильно визначити часовий пояс. Команда `tzselect` має бути доступною за замовчуванням у всіх дистрибутивах Linux, оскільки вона надається пакунком, який містить необхідні службові програми, пов'язані з

бібліотекою GNU C.

Команда `tzselect` буде корисною, наприклад, для користувача, який хоче визначити часовий пояс для “São Paulo City” в “Brazil”. `tzselect` починається із запиту макрообласті бажаного розташування:

```
$ tzselect
Please identify a location so that time zone rules can be set correctly.
Please select a continent, ocean, "coord", or "TZ".
 1) Africa
 2) Americas
 3) Antarctica
 4) Asia
 5) Atlantic Ocean
 6) Australia
 7) Europe
 8) Indian Ocean
 9) Pacific Ocean
10) coord - I want to use geographical coordinates.
11) TZ - I want to specify the time zone using the Posix TZ format.
#? 2
```

Варіант `2` призначений для місць (Північної та Південної) Америки, не обов’язково в одному часовому поясі. Також можна вказати часовий пояс за допомогою географічних координат або нотації зсуву, також відомої як *формат Posix TZ*. Наступним кроком буде вибір країни:

```
Please select a country whose clocks agree with yours.
 1) Anguilla              19) Dominican Republic    37) Peru
 2) Antigua & Barbuda     20) Ecuador               38) Puerto Rico
 3) Argentina             21) El Salvador           39) St Barthelemy
 4) Aruba                 22) French Guiana         40) St Kitts & Nevis
 5) Bahamas               23) Greenland             41) St Lucia
 6) Barbados              24) Grenada               42) St Maarten (Dutch)
 7) Belize                25) Guadeloupe            43) St Martin (French)
 8) Bolivia               26) Guatemala             44) St Pierre & Miquelon
 9) Brazil                27) Guyana                45) St Vincent
10) Canada                28) Haiti                 46) Suriname
11) Caribbean NL          29) Honduras              47) Trinidad & Tobago
12) Cayman Islands        30) Jamaica               48) Turks & Caicos Is
13) Chile                 31) Martinique            49) United States
14) Colombia              32) Mexico                50) Uruguay
```

```
15) Costa Rica           33) Montserrat           51) Venezuela
16) Cuba                 34) Nicaragua            52) Virgin Islands (UK)
17) Curaçao              35) Panama               53) Virgin Islands (US)
18) Dominica             36) Paraguay
#? 9
```

Територія Бразилії охоплює чотири часові пояси, тому лише інформації про країну недостатньо для встановлення часового поясу. На наступному кроці `tzselect` вимагатиме від користувача вказати локальний регіон:

```
Please select one of the following time zone regions.
 1) Atlantic islands
 2) Pará (east); Amapá
 3) Brazil (northeast: MA, PI, CE, RN, PB)
 4) Pernambuco
 5) Tocantins
 6) Alagoas, Sergipe
 7) Bahia
 8) Brazil (southeast: GO, DF, MG, ES, RJ, SP, PR, SC, RS)
 9) Mato Grosso do Sul
10) Mato Grosso
11) Pará (west)
12) Rondônia
13) Roraima
14) Amazonas (east)
15) Amazonas (west)
16) Acre
#? 8
```

Не всі назви населених пунктів доступні, але вибрати найближчий регіон буде достатньо. Надана інформація буде використана `tzselect` для відображення відповідного часового поясу:

```
The following information has been given:

        Brazil
        Brazil (southeast: GO, DF, MG, ES, RJ, SP, PR, SC, RS)

Therefore TZ='America/Sao_Paulo' will be used.
Selected time is now:   sex out 18 18:47:07 -03 2019.
Universal Time is now:  sex out 18 21:47:07 UTC 2019.
Is the above information OK?
```

```
1) Yes
2) No
#? 1

You can make this change permanent for yourself by appending the line
        TZ='America/Sao_Paulo'; export TZ
to the file '.profile' in your home directory; then log out and log in again.

Here is that TZ value again, this time on standard output so that you
can use the /usr/bin/tzselect command in shell scripts:
America/Sao_Paulo
```

Отриману назву часового поясу, `America/Sao_Paulo`, також можна використовувати як вміст `/etc/timezone`, щоб повідомити часовий пояс за замовчуванням для системи:

```
$ cat /etc/timezone
America/Sao_Paulo
```

Як зазначено в результатах `tzselect`, змінна середовища `TZ` визначає часовий пояс для сеансу оболонки, яким би не був часовий пояс системи за замовчуванням. Додавання рядка `TZ='America/Sao_Paulo'; export TZ` до файлу `~/.profile` зробить `America/Sao_Paulo` часовим поясом для майбутніх сеансів користувача. Змінну `TZ` можна також тимчасово змінити під час поточного сеансу, щоб відображати час в іншому часовому поясі:

```
$ env TZ='Africa/Cairo' date
Mon Oct 21 15:45:21 EET 2019
```

У прикладі команда `env` запустить дану команду в новому сеансі суб-оболонки з тими самими змінними середовища поточного сеансу, за винятком змінної `TZ`, зміненої аргументом `TZ='Africa/Cairo'`.

Літній час

Багато регіонів переходять на літній час для частини року, коли годинники зазвичай переводять на годину. Це може призвести до того, що неправильно налаштована система повідомляє неправильний час у цю пору року.

Файл `/etc/localtime` містить дані, які використовуються операційною системою для відповідного налаштування годинника. Стандартні системи Linux мають файли для всіх

часових поясів у каталозі `/usr/share/zoneinfo/`, тому `/etc/localtime` є лише символьним посиланням на фактичний файл даних у цьому каталозі. Файли в `/usr/share/zoneinfo/` впорядковано за назвою відповідного часового поясу, тому файл даних для часового поясу `America/Sao_Paulo` буде `/usr/share/zoneinfo/America/Sao_Paulo`

Оскільки визначення переходу на літній час можуть змінюватися, важливо оновлювати файли в `/usr/share/zoneinfo/`. Команда оновлення інструменту керування пакунками, що надається дистрибутивом, повинна оновлювати їх щоразу, коли доступна нова версія.

Мова та кодування символів

Системи Linux можуть працювати з великою різноманітністю мов і незахідних кодувань символів, ці визначення відомі як *locales*. Найпростішою конфігурацією локалі є визначення змінної середовища `LANG`, за якою більшість програм оболонки визначають мову для використання.

Вміст змінної `LANG` відповідає формату `ab_CD`, де `ab` — код мови, а `CD` — код регіону. Код мови має відповідати стандарту ISO-639, а код регіону – стандарту ISO-3166. Система, налаштована на використання бразильської португальської мови, наприклад, повинна мати змінну `LANG`, визначену як `pt_BR.UTF-8`:

```
$ echo $LANG
pt_BR.UTF-8
```

Як видно у прикладі вихідних даних, змінна `LANG` також містить кодування символів, призначене для системи. ASCII, скорочення від *Американського стандартного коду для обміну інформацією*, був першим широко використовуваним стандартом кодування символів для електронних комунікацій. Однак, оскільки ASCII має дуже обмежений діапазон доступних числових значень і він заснований на англійському алфавіті, він не містить символів, які використовуються іншими мовами, або розширеного набору неалфавітних символів. Кодування UTF-8 є *стандартом Unicode* для звичайних західних символів, а також багатьох інших нетрадиційних символів. Як заявив *Unicode Consortium*, розробник *Unicode Standard*, цей стандарт слід прийняти за замовчуванням, щоб забезпечити сумісність між комп'ютерними платформами:

> Стандарт Unicode надає унікальний номер кожному символу, незалежно від платформи, пристрою, програми чи мови. Він був прийнятий усіма сучасними постачальниками програмного забезпечення і тепер дозволяє транспортувати дані через багато різних платформ, пристроїв і програм без пошкодження. Підтримка

> Unicode є основою для представлення мов і символів у всіх основних операційних системах, пошукових системах, браузерах, ноутбуках і смартфонах, а також в Інтернеті та Всесвітній павутині (URL-адреси, HTML, XML, CSS, JSON тощо). (...) стандарт Unicode і доступність інструментів, що його підтримують, є одними з найбільш значущих останніх глобальних тенденцій програмного забезпечення.
>
> — The Unicode Consortium, What is Unicode?

Деякі системи все ще можуть використовувати стандарти ISO, такі як стандарт ISO-8859-1, для кодування символів, відмінних від ASCII. Однак такі стандарти кодування символів мають вважатись застарілими на користь стандартів кодування Unicode. Тим не менш, усі основні операційні системи, як правило, приймають стандарт Unicode за замовчуванням.

Параметри загальносистемної мови налаштовуються у файлі `/etc/locale.conf`. Змінна `LANG` та інші змінні, пов'язані з локалізацією, призначаються в цьому файлі як звичайна змінна оболонки, наприклад:

```
$ cat /etc/locale.conf
LANG=pt_BR.UTF-8
```

Користувачі можуть використовувати спеціальну конфігурацію локалі, перевизначивши змінну середовища `LANG`. Це можна зробити лише для поточного сеансу або для майбутніх сеансів, додавши нове визначення до профілю Bash користувача в `~/.bash_profile` або `~/.profile`. Однак, доки користувач не ввійде в систему, незалежні від користувача програми, наприклад, екран входу диспетчера дисплеїв, використовуватимуть системну мову за замовчуванням.

TIP Команда `localectl`, доступна в системах, що використовують системний менеджер *systemd*, також може бути використана для запиту та зміни локалі системи. Наприклад: `localectl set-locale LANG=en_US.UTF-8`.

Окрім змінної `LANG`, інші змінні середовища впливають на певні аспекти локалі, наприклад, який символ валюти використовувати або правильний роздільник тисяч для чисел:

`LC_COLLATE`

Встановлює алфавітний порядок. Однією з його цілей є визначення порядку виведення файлів і каталогів.

`LC_CTYPE`

Встановлює, як система оброблятиме певні набори символів. Вона визначає, наприклад,

які символи вважати *верхнім* або *нижнім* регістром.

`LC_MESSAGES`

Встановлює мову для відображення програмних повідомлень (переважно програм GNU).

`LC_MONETARY`

Встановлює грошову одиницю та формат валюти.

`LC_NUMERIC`

Встановлює числовий формат для негрошових значень. Її головна мета — визначити роздільники тисяч і десяткових дробів.

`LC_TIME`

Встановлює формат часу та дати.

`LC_PAPER`

Встановлює стандартний розмір паперу.

`LC_ALL`

Перевизначає всі інші змінні, включаючи `LANG`.

Команда `locale` покаже всі визначені змінні в поточній конфігурації локалі:

```
$ locale
LANG=pt_BR.UTF-8
LC_CTYPE="pt_BR.UTF-8"
LC_NUMERIC=pt_BR.UTF-8
LC_TIME=pt_BR.UTF-8
LC_COLLATE="pt_BR.UTF-8"
LC_MONETARY=pt_BR.UTF-8
LC_MESSAGES="pt_BR.UTF-8"
LC_PAPER=pt_BR.UTF-8
LC_NAME=pt_BR.UTF-8
LC_ADDRESS=pt_BR.UTF-8
LC_TELEPHONE=pt_BR.UTF-8
LC_MEASUREMENT=pt_BR.UTF-8
LC_IDENTIFICATION=pt_BR.UTF-8
LC_ALL=
```

Єдиною невизначеною змінною є `LC_ALL`, яку можна використовувати для тимчасового перевизначення всіх інших налаштувань мови. У наведеному нижче прикладі показано,

як команда `date`, що виконується в системі, налаштованій на мову `pt_BR.UTF-8`, змінюватиме свої вихідні дані відповідно до нової змінної `LC_ALL`:

```
$ date
seg out 21 10:45:21 -03 2019
$ env LC_ALL=en_US.UTF-8 date
Mon Oct 21 10:45:21 -03 2019
```

Модифікація змінної `LC_ALL` дозволила відображати скорочення дня тижня та назви місяця американською англійською мовою (`en_US`). Однак не обов'язково встановлювати однакові локалі для всіх змінних. Можна, наприклад, визначити мову як `pt_BR`, а числовий формат (`LC_NUMERIC`) встановити відповідно до американського стандарту.

Деякі параметри локалізації змінюють спосіб роботи програм з алфавітним упорядкуванням і форматами чисел. У той час як звичайні програми зазвичай готові правильно вибирати загальну мову для таких ситуацій, сценарії можуть поводитися неочікувано, коли, наприклад, намагаються правильно впорядкувати список елементів в алфавітному порядку. З цієї причини рекомендується встановити змінну середовища `LANG` на загальну мову `C`, як у `LANG=C`, щоб сценарій виробляв однозначні результати, незалежно від визначень локалізації, що використовуються в системі, де він виконується. Локаль C проводить лише просте побайтове порівняння, тому вона також працюватиме краще, ніж інші.

Конвертація кодування

Текст може відображатися з незрозумілими символами, якщо він відображається в системі з конфігурацією кодування символів, відмінною від системи, де було створено текст. Щоб вирішити цю проблему, можна використати команду `iconv` шляхом перетворення файлу з оригінального кодування символів на потрібне. Наприклад, щоб конвертувати файл під назвою `original.txt` із кодування ISO-8859-1 у файл під назвою `converted.txt` за допомогою кодування UTF-8, можна використати таку команду:

```
$ iconv -f ISO-8859-1 -t UTF-8 original.txt > converted.txt
```

Опція `-f ISO-8859-1` (або `--from-code=ISO-8859-1`) встановлює кодування вихідного файлу, а опція `-t UTF-8` (або `--to- code=UTF-8`) встановлює кодування для конвертованого файлу. Усі кодування, які підтримуються командою `iconv`, виводяться за допомогою команди `iconv -l` або `iconv --list`. Замість використання переспрямування виведення, як у прикладі, також можна використовувати опцію `-o converted.txt` або

`--output converted.txt`.

Вправи до посібника

1. Виходячи з наступного результату команди `date`, який часовий пояс системи в нотації GMT?

```
$ date
Mon Oct 21 18:45:21 +05 2019
```

2. На який файл має вказувати символьне посилання `/etc/localtime`, щоб зробити `Europe/Brussel` місцевим часом системи за замовчуванням?

3. Символи в текстових файлах можуть не відтворюватися належним чином у системі з кодуванням символів, відмінним від того, що використовується в текстовому документі. Як `iconv` можна використати для перетворення файлу `old.txt` у кодуванні WINDOWS-1252 у файл `new.txt` в кодуванні UTF-8?

Дослідницькі вправи

1. Яка команда зробить `Pacific/Auckland` часовим поясом за умовчанням для поточного сеансу оболонки?

2. Команда `uptime` показує, серед іншого, *середнє завантаження* системи в дробових числах. Вона використовує поточні параметри мови, щоб вирішити, чи має бути роздільником десяткових знаків крапка чи кома. Якщо, наприклад, для поточної мови встановлено `de_DE.UTF-8` (стандартна мова Німеччини), `uptime` використовуватиме кому як роздільник. Знаючи, що в американській англійській мові крапка використовується як роздільник, яка команда змусить `uptime` відображати дроби з використанням крапки замість коми протягом решти поточного сеансу?

3. Команда `iconv` замінить усі символи за межами цільового набору символів знаком питання. Якщо до цільового кодування додано `//TRANSLIT`, символи, яких немає в цільовому наборі символів, буде замінено (транслітеровано) одним або декількома символами схожого вигляду. Як цей метод можна використати для перетворення текстового файлу UTF-8 під назвою `readme.txt` у звичайний файл ASCII під назвою `ascii.txt`?

Підсумки

У цьому уроці описано, як налаштувати систему Linux для роботи з користувацькими мовами та параметрами часу. Також розглядаються поняття та параметри кодування символів, оскільки вони дуже важливі для правильного відтворення текстового вмісту. На уроці розглядаються такі теми:

- Як системи Linux вибирають мову для відображення повідомлень оболонки.
- Розуміння того, як часові пояси впливають на місцевий час.
- Як визначити відповідний часовий пояс і відповідно змінити налаштування системи.
- Що таке кодування символів і як проводити конвертацію між ними.

Розглянуті команди та процедури:

- Змінні середовища, пов'язані з локалізацією та часом, наприклад `LC_ALL`, `LANG` і `TZ`.
- `/etc/timezone`
- `/etc/localtime`
- `/usr/share/zoneinfo/`
- `locale`
- `tzselect`
- `timedatectl`
- `date`
- `iconv`

Відповіді до вправ посібника

1. Виходячи з наступного результату команди `date`, який часовий пояс системи в нотації GMT?

   ```
   $ date
   Mon Oct 21 18:45:21 +05 2019
   ```

   Це часовий пояс `Etc/GMT+5`.

2. На який файл має вказувати символьне посилання `/etc/localtime`, щоб зробити `Europe/Brussel` місцевим часом системи за замовчуванням?

   Посилання `/etc/localtime` має вказувати на `/usr/share/zoneinfo/Europe/Brussels`.

3. Символи в текстових файлах можуть не відтворюватися належним чином у системі з кодуванням символів, відмінним від того, що використовується в текстовому документі. Як `iconv` можна використати для перетворення файлу `old.txt` у кодуванні WINDOWS-1252 у файл `new.txt` в кодуванні UTF-8?

   Команда `iconv -f WINDOWS-1252 -t UTF-8 -o new.txt old.txt` виконає бажане перетворення.

Відповіді до дослідницьких вправ

1. Яка команда зробить `Pacific/Auckland` часовим поясом за умовчанням для поточного сеансу оболонки?

   `export TZ=Pacific/Auckland`

2. Команда `uptime` показує, серед іншого, *середнє завантаження* системи в дробових числах. Вона використовує поточні параметри мови, щоб вирішити, чи має бути роздільником десяткових знаків крапка чи кома. Якщо, наприклад, для поточної мови встановлено `de_DE.UTF-8` (стандартна мова Німеччини), `uptime` використовуватиме кому як роздільник. Знаючи, що в американській англійській мові крапка використовується як роздільник, яка команда змусить `uptime` відображати дроби з використанням крапки замість коми протягом решти поточного сеансу?

   Команда `export LC_NUMERIC=en_US.UTF-8` або `export LC_ALL=en_US.UTF-8`.

3. Команда `iconv` замінить усі символи за межами цільового набору символів знаком питання. Якщо до цільового кодування додано `//TRANSLIT`, символи, яких немає в цільовому наборі символів, буде замінено (транслітеровано) одним або декількома символами схожого вигляду. Як цей метод можна використати для перетворення текстового файлу UTF-8 під назвою `readme.txt` у звичайний файл ASCII під назвою `ascii.txt`?

   Команда `iconv -f UTF-8 -t ASCII//TRANSLIT -o ascii.txt readme.txt` виконає потрібне перетворення.

Linux
Professional
Institute

Розділ 108: Основні системні служби

108.1 Основні системні служби

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 108.1

Ваговий коефіцієнт

3

Ключові галузі знань

- Встановлення системної дати та часу.
- Встановлення апаратного годиннику на правильний час у UTC.
- Налаштування правильного часового поясу.
- Базове налаштування NTP за допомогою ntpd і chrony.
- Вміння користуватися сервісом pool.ntp.org.
- Вміння використовувати команду ntpq.

Частковий список файлів, термінів та утиліт, що використовуються

- `/usr/share/zoneinfo/`
- `/etc/timezone`
- `/etc/localtime`
- `/etc/ntp.conf`
- `/etc/chrony.conf`
- `date`
- `hwclock`
- `timedatectl`

- `ntpd`
- `ntpdate`
- `chronyc`
- `pool.ntp.org`

Linux Professional Institute

108.1 Урок 1

Сертифікат:	LPIC-1 (102)
Версія:	5.0
Розділ:	108 Основні системні служби
Тема:	108.1 Підтримка системного часу
Урок:	1 з 2

Вступ

Точний відлік часу є надзвичайно важливим для сучасної комп'ютерної техніки. Однак реалізація підтримки часу є напрочуд складною. Практика ведення часу здається кінцевому користувачеві тривіальною, але система повинна вміти розумно справлятися з багатьма особливостями та граничними випадками. Майте на увазі, що часові пояси не є статичними, а можуть бути змінені адміністративним чи політичним рішенням. Країна може відмовитися від переходу на літній час. Будь-яка програма повинна вміти логічно обробляти ці зміни. На щастя для системних адміністраторів, рішення для обліку часу в операційній системі Linux є зрілими, надійними і зазвичай працюють без особливих перешкод.

Коли комп'ютер Linux завантажується, він починає відраховувати час. Ми називаємо це *системним годинником*, оскільки він оновлюється операційною системою. Крім того, сучасні комп'ютери також мають *апаратний* або *годинник реального часу*. Цей апаратний годинник часто є функцією материнської плати і зберігає час незалежно від того, працює комп'ютер чи ні. Під час завантаження системний час встановлюється з апаратного годинника, але здебільшого ці два годинники працюють незалежно один від одного. У цьому уроці ми обговоримо методи взаємодії як із системним, так і з апаратним

годинниками.

У більшості сучасних систем Linux системний і апаратний час синхронізуються з *мережевим часом*, який реалізується *протоколом мережевого часу* (NTP). У переважній більшості випадків єдина конфігурація, яку потребує звичайний користувач, це встановити свій часовий пояс, а NTP подбає про все інше. Однак ми розглянемо деякі способи роботи з часом вручну, а особливості налаштування мережевого часу будуть розглянуті в наступному уроці.

Місцевий та всесвітній час

Системний годинник налаштовано на всесвітній координований час (UTC), який є місцевим часом у Гринвічі, Великобританія. Зазвичай користувач хоче знати свій *місцевий час*. Місцевий час обчислюється шляхом використання часу UTC і застосування *зміщення* на основі часового поясу та літнього часу. Таким чином можна уникнути багатьох складнощів.

Системний годинник можна встановити на час UTC або місцевий час, але рекомендується також налаштувати час UTC.

Дата

`date` — основна утиліта, яка просто друкує місцевий час:

```
$ date
Sun Nov 17 12:55:06 EST 2019
```

Зміна параметрів команди `date` змінить формат виведення.

Наприклад, користувач може використовувати `date -u`, щоб переглянути поточний час UTC.

```
$ date -u
Sun Nov 17 18:02:51 UTC 2019
```

Деякі інші параметри, що часто використовуються, повертають місцевий час у форматі, який відповідає прийнятому формату RFC:

`-I`

Дата/час у форматі ISO 8601. Додавання `date` (`-Idate`) обмежить вихідні дані лише

датою. Інші формати: `hours`, `minutes`, `seconds` та `ns` для наносекунд.

`-R`

Повертає дату й час у форматі RFC 5322.

`--rfc-3339`

Повертає дату й час у форматі RFC 3339.

Формат `date` може бути налаштований користувачем за допомогою послідовностей, указаних на man-сторінці. Наприклад, поточний час можна відформатувати як час Unix таким чином:

```
$ date +%s
1574014515
```

З man-сторінки `date` ми бачимо, що `%s` відноситься до часу Unix.

Час Unix використовується в більшості Unix-подібних систем. Він зберігає час UTC як кількість секунд після *Epoch*, яка була визначена як 1 січня 1970 року.

NOTE

Кількість бітів, необхідних для зберігання часу Unix на даний момент, становить 32 біти. У майбутньому виникне проблема, коли 32 біти стануть недостатніми, щоб містити поточний час у форматі Unix. Це спричинить серйозні проблеми для будь-якої 32-розрядної системи Linux. На щастя, це станеться не раніше 19 січня 2038 року.

Використовуючи ці послідовності, ми можемо відформатувати дату й час майже в будь-якому форматі, який вимагає будь-яка програма. Звичайно, у більшості випадків набагато краще дотримуватися прийнятого стандарту.

Крім того, `date --date` можна використовувати для форматування часу, який не є поточним. У цьому сценарії користувач може вказати дату, яка буде застосована до системи, використовуючи, наприклад, час Unix:

```
$ date --date='@1564013011'
Wed Jul 24 20:03:31 EDT 2019
```

Використання параметра `--debug` може бути дуже корисним для забезпечення успішного аналізу дати. Подивіться, що відбувається, коли команді передано дійсну дату:

```
$ date --debug --date="Fri, 03 Jan 2020 14:00:17 -0500"
date: parsed day part: Fri (day ordinal=0 number=5)
date: parsed date part: (Y-M-D) 2020-01-03
date: parsed time part: 14:00:17 UTC-05
date: input timezone: parsed date/time string (-05)
date: using specified time as starting value: '14:00:17'
date: warning: day (Fri) ignored when explicit dates are given
date: starting date/time: '(Y-M-D) 2020-01-03 14:00:17 TZ=-05'
date: '(Y-M-D) 2020-01-03 14:00:17 TZ=-05' = 1578078017 epoch-seconds
date: timezone: system default
date: final: 1578078017.000000000 (epoch-seconds)
date: final: (Y-M-D) 2020-01-03 19:00:17 (UTC)
date: final: (Y-M-D) 2020-01-03 14:00:17 (UTC-05)
```

Це може бути зручним інструментом під час усунення несправностей програми, яка генерує дату.

Апаратний годинник

Користувач може запустити команду `hwclock`, щоб переглянути час, який підтримується на годиннику реального часу. Для цієї команди будуть потрібні підвищені привілеї, тому в цьому випадку ми використаємо `sudo` для виклику команди:

```
$ sudo hwclock
2019-11-20 11:31:29.217627-05:00
```

Використання параметра `--verbose` поверне більше результатів, які можуть бути корисними для усунення несправностей:

```
$ sudo hwclock --verbose
hwclock from util-linux 2.34
System Time: 1578079387.976029
Trying to open: /dev/rtc0
Using the rtc interface to the clock.
Assuming hardware clock is kept in UTC time.
Waiting for clock tick...
...got clock tick
Time read from Hardware Clock: 2020/01/03 19:23:08
Hw clock time : 2020/01/03 19:23:08 = 1578079388 seconds since 1969
Time since last adjustment is 1578079388 seconds
Calculated Hardware Clock drift is 0.000000 seconds
```

```
2020-01-03 14:23:07.948436-05:00
```

Зверніть увагу на `Calculated Hardware Clock drift` (розраховане відхилення апаратного годинника). Ці вихідні дані можуть повідомити вам, чи системний час і апаратний час відрізняються один від одного.

timedatectl

`timedatectl` — це команда, яку можна використовувати для перевірки загального стану часу та дати, включно з тим, чи синхронізовано мережевий час (протокол мережевого часу буде розглянуто в наступному уроці).

За умовчанням `timedatectl` повертає інформацію, подібну до `date`, але з додаванням RTC (апаратного) часу, а також статусу служби NTP:

```
$ timedatectl
               Local time: Thu 2019-12-05 11:08:05 EST
           Universal time: Thu 2019-12-05 16:08:05 UTC
                 RTC time: Thu 2019-12-05 16:08:05
                Time zone: America/Toronto (EST, -0500)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Встановлення часу за допомогою timedatectl

Якщо NTP недоступний, для встановлення часу рекомендується використовувати `timedatectl` замість `date` або `hwclock`:

```
# timedatectl set-time '2011-11-25 14:00:00'
```

Процес подібний до `date`. Користувач також може встановити час незалежно від дати, використовуючи формат HH:MM:SS.

Встановлення часового поясу за допомогою timedatectl

`timedatectl` є кращим способом встановлення місцевого часового поясу в системах Linux на основі `systemd`, коли немає графічного інтерфейсу користувача. `timedatectl` перелічить можливі часові пояси, а потім часовий пояс можна встановити, використовуючи один із них як аргумент.

Спочатку ми виведемо перелік можливих часових поясів:

```
$ timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Algiers
Africa/Bissau
Africa/Cairo
...
```

Список можливих часових поясів довгий, тому в цьому випадку рекомендується використовувати команду `grep`.

Далі ми можемо встановити часовий пояс за допомогою одного з елементів списку, який було повернуто:

```
$ timedatectl set-timezone Africa/Cairo
$ timedatectl
               Local time: Thu 2019-12-05 18:18:10 EET
           Universal time: Thu 2019-12-05 16:18:10 UTC
                 RTC time: Thu 2019-12-05 16:18:10
                Time zone: Africa/Cairo (EET, +0200)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Майте на увазі, що назва часового поясу має бути точною. Наприклад, `Africa/Cairo` змінить часовий пояс, а `cairo` або `africa/cairo` - ні.

Вимкнення NTP за допомогою timedatectl

У деяких випадках може знадобитися відключити NTP. Це можна зробити за допомогою `systemctl`, але ми продемонструємо це за допомогою `timedatectl`:

```
# timedatectl set-ntp no
$ timedatectl
             Local time: Thu 2019-12-05 18:19:04 EET Universal time: Thu 2019-12-05 16:19:04
UTC
               RTC time: Thu 2019-12-05 16:19:04
              Time zone: Africa/Cairo (EET, +0200)
            NTP enabled: no
```

```
       NTP synchronized: no
        RTC in local TZ: no
             DST active: n/a
```

Налаштування часового поясу без timedatectl

Встановлення інформації про часовий пояс є стандартним кроком під час встановлення Linux на нову машину. Якщо є графічний процес інсталяції, це, швидше за все, буде виконано без участі користувача.

Каталог `/usr/share/zoneinfo` містить інформацію про різні можливі часові пояси. У каталозі `zoneinfo` є підкаталоги, які містять назви континентів, а також інші символьні посилання. Рекомендовано шукати інформацію про зону вашого регіону, починаючи з вашого континенту.

Файли `zoneinfo` містять правила, необхідні для розрахунку зміщення місцевого часу відносно UTC, і вони також важливі, якщо у вашому регіоні дотримуються літнього часу. Вміст `/etc/localtime` буде прочитано, коли Linux потрібно буде визначити місцевий часовий пояс. Щоб установити часовий пояс без використання графічного інтерфейсу користувача, користувач повинен створити символьне посилання для свого місцезнаходження з `/usr/share/zoneinfo` на `/etc/localtime`. Наприклад:

```
$ ln -s /usr/share/zoneinfo/Canada/Eastern /etc/localtime
```

Після встановлення правильного часового поясу часто рекомендується виконати:

```
# hwclock --systohc
```

Це встановить *апаратний годинник* із *системного годинника* (тобто годинник реального часу буде встановлено на той самий час, що й за допомогою команди `date`). Будь ласка, зверніть увагу, що ця команда виконується з правами root, у цьому випадку, якщо ви ввійшли в систему як root.

`/etc/timezone` схожий на `/etc/localtime`. Це представлення даних місцевого часового поясу, і його можна прочитати за допомогою `cat`:

```
$ cat /etc/timezone
America/Toronto
```

Зауважте, що цей файл використовується не всіма дистрибутивами Linux.

Встановлення дати й часу без timedatectl

NOTE

Більшість сучасних систем Linux використовують `systemd` для конфігурації та сервісів, тому не рекомендується використовувати `date` або `hwclock` для встановлення часу. `systemd` використовує `timedatectl` для цього. Тим не менш, важливо знати ці застарілі команди, якщо вам потрібно буде адмініструвати старішу систему.

Використання дати

`date` має опцію для встановлення системного часу. Використовуйте `--set` або `-s`, щоб встановити дату та час. Ви також можете використовувати `--debug`, щоб перевірити правильність аналізу команди:

```
# date --set="11 Nov 2011 11:11:11"
```

Зверніть увагу, що для встановлення дати тут потрібні права root. Ми також можемо змінити час або дату окремо:

```
# date +%Y%m%d -s "20111125"
```

Тут ми повинні вказати послідовності, щоб наш рядок аналізувався належним чином. Наприклад, `%Y` стосується року, тому перші чотири цифри `2011` будуть інтерпретуватися як 2011 рік. Подібним чином, `%T` — це послідовність часу, і це демонструється тут за допомогою встановлення часу:

```
# date +%T -s "13:11:00"
```

Після зміни системного часу рекомендується також налаштувати апаратний годинник, щоб системний і апаратний годинник були синхронізовані:

```
# hwclock --systohc
```

`systohc` означає “системний годинник для апаратного годинника”.

Використання hwclock

Замість того, щоб налаштовувати системний годинник і оновлювати апаратний годинник, ви можете змінити процес у зворотному напрямку. Ми почнемо з налаштування апаратного годинника:

```
# hwclock --set --date "4/12/2019 11:15:19"
# hwclock
Fri 12 Apr 2019 6:15:19 AM EST -0.562862 seconds
```

Зауважте, що за замовчуванням `hwclock` очікує часу UTC, але за замовчуванням повертає місцевий час.

Після встановлення апаратного годинника нам потрібно буде оновити системний годинник з нього. `hctosys` можна розуміти як “апаратний годинник для системного годинника”.

```
# hwclock --hctosys
```

Вправи до посібника

1. Укажіть, чи наступні команди відображають або змінюють *системний час* або *апаратний час*:

Команда(и)	Системний час	Апаратний час	Обидва варіанти
`date -u`			
`hwclock --set --date "12:00:00"`			
`timedatectl`			
`timedatectl | grep RTC`			
`hwclock --hctosys`			
`date +%T -s "08:00:00"`			
`timedatectl set-time 1980-01-10`			

2. Зверніть увагу на наведені нижче вихідні дані, а потім виправте формат аргументу, щоб команда була успішною:

```
$ date --debug --date "20/20/12 0:10 -3"

date: warning: value 20 has less than 4 digits. Assuming MM/DD/YY[YY]
date: parsed date part: (Y-M-D) 0002-20-20
date: parsed time part: 00:10:00 UTC-03
date: input timezone: parsed date/time string (-03)
date: using specified time as starting value: '00:10:00'
date: error: invalid date/time value:
date:     user provided time: '(Y-M-D) 0002-20-20 00:10:00 TZ=-03'
date:        normalized time: '(Y-M-D) 0003-08-20 00:10:00 TZ=-03'
date:                                 ---- --
date:      possible reasons:
date:        numeric values overflow;
date:        incorrect timezone
date: invalid date ‘20/20/2 0:10 -3’
```

3. Використовуйте команду `date` і послідовності, щоб місяцем системи було встановлено лютий. Залиште решту дати й часу без змін.

4. Якщо припустити, що попередня команда виконана успішно, використовуйте `hwclock`, щоб встановити апаратний годинник із системного годинника.

5. Є місце під назвою `eucla`. До складу якого материка воно входить? Використовуйте команду `grep`, щоб дізнатися.

6. Встановіть поточний часовий пояс для `eucla`.

Дослідницькі вправи

1. Який спосіб встановлення часу є оптимальним? За якого сценарію бажаний метод може бути неможливим?

2. Чому, на вашу думку, існує так багато методів, щоб виконати одну й ту саму дію, наприклад встановити системний час?

3. Після 19 січня 2038 року системний час Linux потребуватиме для зберігання 64-розрядного числа. Однак можливо, що ми могли б просто вибрати “New Epoch” (Нову Епоху). Наприклад, опівночі 1 січня 2038 року можна встановити час нової епохи `0`. Чому, на вашу думку, це рішення не стало кращим?

Підсумки

На цьому уроці ви вивчили:

- Як відображати час у різних форматах з командного рядка.
- Різницю між системним годинником і апаратним годинником у Linux.
- Як вручну встановити системний годинник.
- Як вручну встановити апаратний годинник.
- Як змінити часовий пояс системи.

Команди, які використовуються в цьому уроці:

`date`
: Відображає або змінює час системного годинника. Інші варіанти:

  `-u`
  : Відображає час UTC.

  `+%s`
  : Використовує послідовність для відображення часу епохи.

  `--date=`
  : Вказує конкретний час для відображення на відміну від поточного часу.

  `--debug`
  : Відображає повідомлення налагодження під час аналізу дати, введеної користувачем.

  `-s`
  : Налаштовує системний годинник вручну.

`hwclock`
: Відображає або змінює час апаратного годинника.

  `--systohc`
  : Використовує системний годинник, щоб установити апаратний годинник.

  `--hctosys`
  : Використовує апаратний годинник для встановлення системного годинника.

`--set --date`

Налаштовує апаратний годинник вручну.

`timedatectl`

Відображає системний і апаратний годинники, а також конфігурації NTP у системах Linux на основі systemd.

`set-time`

Встановлює час вручну.

`list-timezones`

Перелічує можливі часові пояси.

`set-timezone`

Встановлює часові зони вручну.

`set-ntp`

Вмикає/вимикає NTP.

Відповіді до вправ посібника

1. Укажіть, чи наступні команди відображають або змінюють *системний час* або *апаратний час*:

Команда(и)	Системний час	Апаратний час	Обидва варіанти
`date -u`	X		
`hwclock --set --date "12:00:00"`		X	
`timedatectl`			X
`timedatectl | grep RTC`		X	
`hwclock --hctosys`	X		
`date +%T -s "08:00:00"`	X		
`timedatectl set-time 1980-01-10`			X

2. Зверніть увагу на наведені нижче вихідні дані, а потім виправте формат аргументу, щоб команда була успішною:

```
$ date --debug --date "20/20/12 0:10 -3"

date: warning: value 20 has less than 4 digits. Assuming MM/DD/YY[YY]
date: parsed date part: (Y-M-D) 0002-20-20
date: parsed time part: 00:10:00 UTC-03
date: input timezone: parsed date/time string (-03)
date: using specified time as starting value: '00:10:00'
date: error: invalid date/time value:
date:     user provided time: '(Y-M-D) 0002-20-20 00:10:00 TZ=-03'
date:        normalized time: '(Y-M-D) 0003-08-20 00:10:00 TZ=-03'
date:                                 ---- --
date:      possible reasons:
date:        numeric values overflow;
date:        incorrect timezone
date: invalid date ‘20/20/2 0:10 -3’
```

`date --debug --set "12/20/20 0:10 -3"`

3. Використовуйте команду `date` і послідовності, щоб місяцем системи було встановлено лютий. Залиште решту дати й часу без змін.

   ```
   date +%m -s "2"
   ```

4. Якщо припустити, що попередня команда виконана успішно, використовуйте `hwclock`, щоб встановити апаратний годинник із системного годинника.

   ```
   hwclock -systohc
   ```

5. Є місце під назвою `eucla`. До складу якого материка воно входить? Використовуйте команду `grep`, щоб дізнатися.

   ```
   timedatectl list-timezones \| grep -i eucla
   ```

   АБО

   ```
   grep -ri eucla /usr/share/zoneinfo
   ```

6. Встановіть поточний часовий пояс для `eucla`.

   ```
   timedatectl set-timezone 'Australia/Eucla'
   ```

   або

   ```
   ln -s /usr/share/zoneinfo/Australia/Eucla /etc/localtime
   ```

Відповіді до дослідницьких вправ

1. Який спосіб встановлення часу є оптимальним? За якого сценарію бажаний метод може бути неможливим?

   У більшості дистрибутивів Linux NTP увімкнено за замовчуванням і його слід залишити для встановлення системного часу без втручання. Однак, якщо є система Linux, яка не підключена до Інтернету, NTP буде недоступний. Наприклад, вбудована система Linux, яка працює на промисловому обладнанні, може не мати підключення до мережі.

2. Чому, на вашу думку, існує так багато методів, щоб виконати одну й ту саму дію, наприклад встановити системний час?

   Оскільки встановлення часу було вимогою для всіх систем *nix протягом десятиліть, існує багато застарілих методів встановлення часу, які досі підтримуються.

3. Після 19 січня 2038 року системний час Linux потребуватиме для зберігання 64-розрядного числа. Однак можливо, що ми могли б просто вибрати “New Epoch” (Нову Епоху). Наприклад, опівночі 1 січня 2038 року можна встановити час нової епохи `0`. Чому, на вашу думку, це рішення не стало кращим?

   До 2038 року переважна більшість комп’ютерів уже працюватиме з 64-розрядними процесорами, і використання 64-розрядного числа не призведе до істотного зниження продуктивності. Однак оцінити ризики “перезавантаження” епохального часу таким чином було б неможливо. Це може вплинути на багато застарілого програмного забезпечення. Банки та великі підприємства, наприклад, часто мають велику кількість старих програм для внутрішнього використання. Тож цей сценарій, як і багато інших, є дослідженням компромісів. На будь-які 32-розрядні системи, які все ще працюють у 2038 році, вплине переповнення Epoch Time, але на застаріле програмне забезпечення вплине зміна значення Epoch.

108.1 Урок 2

Сертифікат:	LPIC-1 (102)
Версія:	5.0
Розділ:	108 Основні системні служби
Тема:	108.1 Підтримка системного часу
Урок:	2 з 2

Вступ

У той час як персональні комп'ютери здатні підтримувати досить точний час самостійно, виробничі обчислювальні та мережеві середовища вимагають дуже точного ведення часу. Найточніший час вимірюється за допомогою *еталонних годинників*, якими зазвичай є атомні годинники. Сучасний світ розробив систему, у якій усі підключені до Інтернету комп'ютерні системи можна синхронізувати з цими еталонними годинниками за допомогою так званого *протоколу мережевого часу* (NTP). Комп'ютерна система з NTP зможе синхронізувати свої системні годинники з часом, який надають опорні годинники. Якщо системний час і час, виміряний на цих серверах, відрізняються, комп'ютер поступово прискорює або сповільнює внутрішній системний час, доки системний час не збігається з мережевим.

NTP використовує ієрархічну структуру для поширення часу. Еталонні годинники підключені до серверів на вершині ієрархії. Ці сервери є машинами *Stratum 1* і зазвичай до них мамає публічного доступу. Однак машини Stratum 1 доступні для машин *Stratum 2*, які доступні для машин *Stratum 3* і так далі. Сервери Stratum 2 доступні публічно, як і будь-які машини нижчого рівня в ієрархії. Під час налаштування протоколу NTP для великої мережі доцільно мати невелику кількість комп'ютерів, підключених до серверів Stratum

2+, а потім ці машини мають поширювати час через NTP для всіх інших машин. Таким чином можна мінімізувати вимоги до машин Stratum 2.

Є кілька важливих термінів, які виникають під час обговорення NTP. Деякі з цих термінів використовуються в командах, які ми будемо використовувати для відстеження стану NTP на наших машинах:

Offset

Це стосується абсолютної різниці між системним часом і часом NTP. Наприклад, якщо системний годинник показує 12:00:02, а час NTP – 11:59:58, то зміщення між двома годинниками становить чотири секунди.

Step

Якщо часовий зсув між постачальником NTP і споживачем перевищує 128 мс, NTP виконає одну значну зміну системного часу, а не сповільнить або прискорить системний час. Це називається *stepping*.

Slew

Поворотний час стосується змін системного часу, коли зсув між системним часом і NTP становить менше 128 мс. Якщо це так, то зміни відбуватимуться поступово. Це називається *slewing*.

Insane Time

Якщо зсув між системним часом і часом NTP перевищує 17 хвилин, тоді системний час вважається *insane* (божевільним) і демон NTP не вноситиме жодних змін у системний час. Необхідно вжити спеціальних заходів, щоб привести системний час стану в межах 17 хвилин від правильного.

Drift

Дрейф стосується явища, коли два годинники з часом розсинхронізуються. По суті, якщо два годинники спочатку синхронізовані, але потім з часом розсинхронізуються, тоді відбувається дрейф годинника.

Jitter

Джиттер стосується величини дрейфу з часу останнього запиту годинника. Отже, якщо остання синхронізація NTP відбулася 17 хвилин тому, а зміщення між постачальником і споживачем NTP становить 3 мілісекунди, тоді 3 мілісекунди є джиттером.

Тепер ми обговоримо деякі конкретні способи реалізації NTP у Linux.

timedatectl

Якщо ваш дистрибутив Linux використовує `timedatectl`, то за умовчанням він реалізує клієнт *SNTP*, а не повну реалізацію NTP. Це менш складна реалізація мережевого часу, яка означає, що ваш комп'ютер не обслуговуватиме NTP для інших підключених комп'ютерів.

У цьому випадку SNTP не працюватиме, якщо не запущено службу `timesyncd`. Як і з усіма службами systemd, ми можемо перевірити, що вони працюють за допомогою:

```
$ systemctl status systemd-timesyncd
● systemd-timesyncd.service - Network Time Synchronization
   Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; enabled; vendor preset:
enabled)
  Drop-In: /lib/systemd/system/systemd-timesyncd.service.d
           └─disable-with-time-daemon.conf
   Active: active (running) since Thu 2020-01-09 21:01:50 EST; 2 weeks 1 days ago
     Docs: man:systemd-timesyncd.service(8)
 Main PID: 1032 (systemd-timesyn)
   Status: "Synchronized to time server for the first time 91.189.89.198:123
(ntp.ubuntu.com)."
    Tasks: 2 (limit: 4915)
   Memory: 3.0M
   CGroup: /system.slice/systemd-timesyncd.service
           └─1032 /lib/systemd/systemd-timesyncd

Jan 11 13:06:18 NeoMex systemd-timesyncd[1032]: Synchronized to time server for the first
time 91.189.91.157:123 (ntp.ubuntu.com).
...
```

Статус синхронізації SNTP `timedatectl` можна перевірити за допомогою `show-timesync`:

```
$ timedatectl show-timesync --all
LinkNTPServers=
SystemNTPServers=
FallbackNTPServers=ntp.ubuntu.com
ServerName=ntp.ubuntu.com
ServerAddress=91.189.89.198
RootDistanceMaxUSec=5s
PollIntervalMinUSec=32s
PollIntervalMaxUSec=34min 8s
PollIntervalUSec=34min 8s
NTPMessage={ Leap=0, Version=4, Mode=4, Stratum=2, Precision=-23, RootDelay=8.270ms,
```

```
RootDispersion=18.432ms, Reference=91EECB0E, OriginateTimestamp=Sat 2020-01-25 18:35:49 EST,
ReceiveTimestamp=Sat 2020-01-25 18:35:49 EST, TransmitTimestamp=Sat 2020-01-25 18:35:49 EST,
DestinationTimestamp=Sat 2020-01-25 18:35:49 EST, Ignored=no PacketCount=263, Jitter=2.751ms
}
Frequency=-211336
```

Ця конфігурація може бути адекватною для більшості ситуацій, але, як зазначалося раніше, її буде недостатньо, якщо хтось сподівається синхронізувати кілька клієнтів у мережі. У цьому випадку рекомендується встановити повний клієнт NTP.

NTP-демон

Системний час порівнюється з часом мережі на регулярній основі. Щоб це працювало, ми повинні мати *демон*, що працює у фоновому режимі. Для багатьох систем Linux ім'я цього демона – `ntpd`. `ntpd` дозволить машині бути не тільки *споживачем часу* (тобто, здатною синхронізувати свій власний годинник із зовнішнього джерела), але також *надавати* час іншим машинам.

Припустимо, що наш комп'ютер базується на systemd і використовує `systemctl` для керування демонами. Ми встановимо пакет `ntp` за допомогою відповідного менеджера пакунків, а потім переконаємося, що наш демон `ntpd` працює, перевіривши його статус:

```
$ systemctl status ntpd

●  ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2019-12-06 03:27:21 EST; 7h ago
  Process: 856 ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 867 (ntpd)
   CGroup: /system.slice/ntpd.service
           `-867 /usr/sbin/ntpd -u ntp:ntp -g
```

У деяких випадках може знадобитися як запустити, так і ввімкнути `ntpd`. На більшості машин Linux це досягається за допомогою:

```
# systemctl enable ntpd && systemctl start ntpd
```

Запити NTP надходять на TCP-порт 123. Якщо NTP не працює, переконайтеся, що цей порт відкритий і прослуховується.

Конфігурація NTP

NTP може опитувати кілька джерел і вибирати найкращих кандидатів для встановлення системного часу. Якщо з'єднання з мережею втрачено, NTP використовує попередні коригування зі своєї історії, щоб оцінити майбутні коригування.

Залежно від вашого дистрибутива Linux, список мережевих серверів часу буде зберігатися в різних місцях. Припустимо, що на вашій машині встановлено `ntp`.

Файл `/etc/ntp.conf` містить конфігураційну інформацію про те, як ваша система синхронізується з мережевим часом. Цей файл можна читати та змінювати за допомогою `vi` або `nano`.

За замовчуванням NTP-сервери, що використовуються, будуть указані в такому розділі:

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
```

Синтаксис додавання NTP-серверів виглядає так:

```
server (IP Address)
server server.url.localhost
```

Адреси серверів можуть бути IP-адресами або URL-адресами, якщо DNS правильно налаштовано. У цьому випадку сервер завжди буде запитуватися.

Адміністратор мережі може також розглянути можливість використання (або налаштування) *пулу*. У цьому випадку ми припускаємо, що є кілька постачальників NTP, які працюють з демонами NTP і мають однаковий час. Коли клієнт запитує пул, постачальник вибирається випадковим чином. Це допомагає розподілити мережеве навантаження між багатьма машинами, щоб жодна машина в пулі не обробляла всі запити NTP.

Зазвичай `/etc/ntp.conf` заповнюється пулом серверів під назвою `pool.ntp.org`. Так, наприклад, `server 0.centos.pool.ntp.org` є пулом NTP за умовчанням, який надається машинам CentOS.

pool.ntp.org

Сервери NTP, які використовуються за замовчуванням, є проектом з відкритим кодом. Більше інформації можна знайти на ntppool.org.

> Подумайте, чи підходить пул NTP вам для використання. Якщо бізнес, організація чи людське життя залежать від правильного часу або можуть постраждати від того, що він неправильний, вам не слід "просто видаляти це з Інтернету". NTP Pool загалом дуже якісний, але це служба, якою керують волонтери у вільний час. Будь ласка, зверніться до своїх постачальників обладнання та сервісів, щоб отримати локальне та надійне обслуговування. Перегляньте також умови обслуговування. Ми рекомендуємо сервери часу від Meinberg, але ви також можете знайти сервери часу від End Run, Spectracom та багатьох інших.
>
> — ntppool.org

ntpdate

Під час початкового налаштування системний час і NTP можуть бути серйозно десинхронізовані. Якщо *зсув* між системним і NTP-часом перевищує 17 хвилин, то NTP-демон не вноситиме зміни в системний час. У цьому сценарії буде потрібно ручне втручання.

По-перше, якщо `ntpd` працює, необхідно буде *зупинити* службу. Для цього використовуйте `systemctl stop ntpd`.

Далі використовуйте `ntpdate pool.ntp.org`, щоб виконати початкову одноразову синхронізацію, де `pool.ntp.org` посилається на IP-адресу або URL-адресу NTP-сервера. Може знадобитися кілька синхронізацій.

ntpq

`ntpq` — утиліта для моніторингу стану NTP. Після запуску та налаштування демона NTP `ntpq` можна використовувати для перевірки його стану:

```
$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
+37.44.185.42    91.189.94.4      3 u   86  128  377  126.509  -20.398   6.838
+ntp2.0x00.lv    193.204.114.233  2 u   82  128  377  143.885   -8.105   8.478
*inspektor-vlan1 121.131.112.137  2 u   17  128  377  112.878  -23.619   7.959
```

```
 b1-66er.matrix. 18.26.4.105      2 u  484  128   10   34.907   -0.811  16.123
```

У цьому випадку `-p` означає *print*, і він виведе перелік вузлів. Адреси хостів також можна повернути як IP-адреси за допомогою `-n`.

`remote`

Ім'я хоста постачальника NTP.

`refid`

Ідентифікатор постачальника NTP.

`st`

Стратум провайдера.

`when`

Кількість секунд після останнього запиту.

`poll`

Кількість секунд між запитами.

`reach`

Ідентифікатор статусу, щоб вказати, чи було досягнуто сервера. Успішні підключення збільшать це число на 1.

`delay`

Час у мс між запитом і відповіддю сервера.

`offset`

Час у мс між системним часом і часом NTP.

`jitter`

Зсув у мс між системним часом і NTP в останньому запиті.

`ntpq` також має інтерактивний режим, до якого можна отримати доступ, коли його запускають без параметрів або аргументів. Параметр `?` поверне список команд, які `ntpq` розпізнає.

chrony

`chrony` — ще один спосіб реалізації NTP. Він встановлений за замовчуванням у деяких

системах Linux, але доступний для завантаження в усіх основних дистрибутивах. `chronyd` — це демон chrony, а `chronyc` — це інтерфейс командного рядка. Може знадобитися запустити та ввімкнути `chronyd` перед взаємодією з `chronyc`.

Якщо встановлення chrony має конфігурацію за замовчуванням, тоді використання команди `chronyc tracking` надасть інформацію про NTP і системний час:

```
$ chronyc tracking
Reference ID    : 3265FB3D (bras-vprn-toroon2638w-lp130-11-50-101-251-61.dsl.)
Stratum         : 3
Ref time (UTC)  : Thu Jan 09 19:18:35 2020
System time     : 0.000134029 seconds fast of NTP time
Last offset     : +0.000166506 seconds
RMS offset      : 0.000470712 seconds
Frequency       : 919.818 ppm slow
Residual freq   : +0.078 ppm
Skew            : 0.555 ppm
Root delay      : 0.006151616 seconds
Root dispersion : 0.010947504 seconds
Update interval : 129.8 seconds
Leap status     : Normal
```

Ці вихідні дані містять багато інформації, більше ніж те, що доступно в інших варіантах.

`Reference ID`

Ідентифікатор та ім’я сервера, з яким зараз синхронізовано комп’ютер.

`Stratum`

Кількість переходів до комп’ютера з підключеним еталонним годинником.

`Ref time`

Це час UTC, коли було зроблено останнє вимірювання з еталонного джерела.

`System time`

Затримка системного годинника від синхронізованого сервера.

`Last offset`

Приблизний зсув останнього оновлення годинника.

`RMS offset`

Довгострокове середнє значення зміщення.

`Frequency`

Це частота, з якою системний годинник буде неправильно працювати, якщо chronyd не виправляє його. Він надається в ppm (частки на мільйон).

`Residual freq`

Залишкова частота, що вказує на різницю між вимірюваннями від еталонного джерела та частотою, що використовується в даний момент.

`Skew`

Розрахункова межа похибки частоти.

`Root delay`

Загальний час затримки мережевого шляху до стратума комп'ютера, з яким комп'ютер синхронізується.

`Leap status`

Це статус стрибка, який може мати одне з наступних значень – нормальний, вставити секунду, видалити секунду або не синхронізовано.

Ми також можемо переглянути детальну інформацію про останнє актуальне оновлення NTP:

```
# chrony ntpdata
Remote address  : 172.105.97.111 (AC69616F)
Remote port     : 123
Local address   : 192.168.122.81 (C0A87A51)
Leap status     : Normal
Version         : 4
Mode            : Server
Stratum         : 2
Poll interval   : 6 (64 seconds)
Precision       : -25 (0.000000030 seconds)
Root delay      : 0.000381 seconds
Root dispersion : 0.000092 seconds
Reference ID    : 61B7CE58 ()
Reference time  : Mon Jan 13 21:50:03 2020
Offset          : +0.000491960 seconds
Peer delay      : 0.004312567 seconds
Peer dispersion : 0.000000068 seconds
Response time   : 0.000037078 seconds
Jitter asymmetry: +0.00
NTP tests       : 111 111 1111
```

```
Interleaved     : No
Authenticated   : No
TX timestamping : Daemon
RX timestamping : Kernel
Total TX        : 15
Total RX        : 15
Total valid RX  : 15
```

Нарешті, `chronyc sources` поверне інформацію про сервери NTP, які використовуються для синхронізації часу:

```
$ chronyc sources
210 Number of sources = 0
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
```

На даний момент на цій машині немає налаштованих джерел. Ми можемо додати джерела з `pool.ntp.org`, відкривши файл конфігурації chrony. Зазвичай він знаходиться за адресою `/etc/chrony.conf`. Коли ми відкриваємо цей файл, ми маємо побачити, що деякі сервери прописані за замовчуванням:

```
210 Number of sources = 0
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
# Most computers using chrony will send measurement requests to one or
# more 'NTP servers'.  You will probably find that your Internet Service
# Provider or company have one or more NTP servers that you can specify.
# Failing that, there are a lot of public NTP servers.  There is a list
# you can access at http://support.ntp.org/bin/view/Servers/WebHome or
# you can use servers from the 3.arch.pool.ntp.org project.

! server 0.arch.pool.ntp.org iburst iburst
! server 1.arch.pool.ntp.org iburst iburst
! server 2.arch.pool.ntp.org iburst iburst

! pool 3.arch.pool.ntp.org iburst
```

Ці сервери також слугуватимуть синтаксичним посібником під час введення наших власних серверів. Однак у цьому випадку ми просто видалимо `!` на початку кожного рядка, таким чином розкоментувавши ці рядки та використавши стандартні сервери з проекту `pool.ntp.org`.

Крім того, у цьому файлі ми можемо змінити конфігурацію за замовчуванням щодо skew (зсуву) та дрейфу, а також розташування дрейфового файлу та ключового файлу.

На цій машині нам потрібно зробити велику початкову корекцію годинника. Ми розкоментуємо наступний рядок:

```
! makestep 1.0 3
```

Після внесення змін у файл конфігурації перезапустіть службу `chronyd`, а потім скористайтеся `chronyc makestep`, щоб вручну змінити системний годинник:

```
# chronyc makestep
200 OK
```

А потім використовуйте `chronyc tracking`, як і раніше, щоб перевірити, чи відбулися зміни.

Вправи до посібника

1. Зазначте відповідний термін для кожного визначення:

Визначення	**Термін**
Комп'ютер, який буде ділитися з вами мережевим часом	
Відстань від контрольного годинника, у стрибках або кроках	
Різниця між системним часом і мережним часом	
Різниця між системним часом і мережним часом з часу останнього опитування NTP	
Група серверів, які забезпечують мережевий час і розподіляють навантаження між собою	

2. Укажіть, яку з команд ви будете використовувати для виведення таких значень:

Value	`chronyc tracking`	`timedatectl show-timesync --all`	`ntpq -pn`	`chrony ntpdata`	`chronyc sources`
Джиттер					
Дрейф					
Інтервал пулу					
Зсув					
Стратум					
IP адреса провайдера					
Root-затримка					

3. Ви налаштовуєте корпоративну мережу, що складається з сервера Linux і кількох

робочих станцій Linux. Сервер має статичну IP-адресу 192.168.0.101. Ви вирішуєте, що сервер підключатиметься до `pool.ntp.org`, а потім надаватиме NTP-час робочим станціям. Опишіть конфігурацію сервера та робочих станцій.

4. Машина Linux має неправильний час. Опишіть кроки, які ви б зробили для усунення NTP-несправностей.

Дослідницькі вправи

1. Дослідіть відмінності між SNTP і NTP.

SNTP	NTP

2. Чому системний адміністратор може *не* використовувати `pool.ntp.org`?

3. Як системний адміністратор може приєднатися до проекту `pool.ntp.org` або зробити інший внесок у нього?

Підсумки

На цьому уроці ви вивчили:

- Що таке NTP і чому це важливо.
- Налаштування демона NTP із проекту `pool.ntp.org`.
- Використання `ntpq` для перевірки конфігурації NTP.
- Використання `chrony` як альтернативної служби NTP.

Команди, які використовуються в цьому уроці:

`timedatectl show-timesync --all`
Відображає інформацію SNTP, якщо використовується `timedatectl`.

`ntpdate <address>`
Виконайте одноразове оновлення кроку NTP вручну.

`ntpq -p`
Роздрукуйте історію останніх опитувань NTP. `-n` замінить URL-адреси IP-адресами.

`chronyc tracking`
Відображає стан NTP, якщо використовується Chrony.

`chronyc ntpdata`
Відображає інформацію NTP про останнє опитування.

`chronyc sources`
Відображає інформацію про постачальників NTP.

`chronyc makestep`
Виконайте ручне одноразове оновлення кроку NTP, якщо використовується chrony.

Відповіді до вправ посібника

1. Введіть відповідний термін для кожного визначення:

Визначення	**Термін**
Комп'ютер, який буде ділитися з вами мережевим часом	Провайдер
Відстань від контрольного годинника, у стрибках або кроках	Стратум
Різниця між системним часом і мережевим часом	Зсув
Різниця між системним часом і часом мережі з часу останнього опитування NTP	Джиттер
Група серверів, які забезпечують мережевий час і розподіляють навантаження між собою	Пул

2. Укажіть, яку з команд ви будете використовувати для виведення таких значень:

Value	`chronyc tracking`	`timedatectl show-timesync --all`	`ntpq -pn`	`chrony ntpdata`	`chronyc sources`
Джиттер		X	X		
Дрейф					
Інтервал пулу l	X	X	X (колонка `when`)	X	X
Зсув	X		X	X	
Стратум	X	X	X	X	X
IP-адреса провайдера		X	X	X	X
Root-затримка	X			X	

3. Ви налаштовуєте корпоративну мережу, що складається з сервера Linux і кількох

робочих станцій Linux. Сервер має статичну IP-адресу 192.168.0.101. Ви вирішуєте, що сервер підключатиметься до `pool.ntp.org`, а потім надаватиме NTP-час робочим станціям. Опишіть конфігурацію сервера та робочих станцій.

Переконайтеся, що на сервері працює служба ntpd, а не SNTP. Використовуйте пули `pool.ntp.org` у файлі `/etc/ntp.conf` або `/etc/chrony.conf`. Для кожного клієнта вкажіть `192.168.0.101` у кожному файлі `/etc/ntp.conf` або `/etc/chrony.conf`.

4. Машина Linux має неправильний час. Опишіть кроки, які ви б зробили для усунення несправностей NTP.

   Спочатку переконайтеся, що комп'ютер підключено до Інтернету. Для цього використовуйте `ping`. Перевірте, чи працює служба ntpd або SNTP за допомогою `systemctl status ntpd` або `systemctl status systemd-timesyncd`. Ви можете бачити повідомлення про помилки, які містять корисну інформацію. Нарешті, скористайтеся такою командою, як `ntpq -p` або `chrony tracking`, щоб перевірити, чи були зроблені будь-які запити. Якщо системний час суттєво відрізняється від мережевого, можливо, системний час вважається "божевільним" і не буде змінено без ручного втручання. У цьому випадку скористайтеся командою з попереднього уроку або такою командою, як `ntpdate pool.ntp.org`, щоб виконати одноразову синхронізацію ntp.

Відповіді до дослідницьких вправ

1. Дослідіть відмінності між SNTP і NTP.

SNTP	**NTP**
менш точні	більш точні
вимагає менше ресурсів	вимагає більше ресурсів
не може діяти як постачальник часу	може діяти як постачальник часу
лише час кроків	кроки або скорочення часу
запитує час з одного джерела	може контролювати кілька серверів NTP і використовувати оптимального провайдера

2. Чому системний адміністратор може *не* використовувати `pool.ntp.org`?

   Від ntppool.org: якщо абсолютно важливо мати правильний час, вам слід розглянути альтернативу. Подібним чином, якщо ваш Інтернет-провайдер має сервер часу, рекомендується використовувати його.

3. Як системний адміністратор може приєднатися до проекту `pool.ntp.org` або зробити інший внесок у нього?

   Від www.ntppool.org: Ваш сервер повинен мати статичну IP-адресу та постійне підключення до Інтернету. Статична IP-адреса не повинна змінюватися взагалі або принаймні рідше одного разу на рік. Окрім цього, вимоги до пропускної здатності є скромними: 384–512 Кбіт. Запрошуємо приєднатися до серверів Stratum 3 або 4.

108.2 Системне журналювання

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 108.2

Ваговий коефіцієнт

4

Ключові галузі знань

- Базова конфігурація rsyslog.
- Розуміння стандартних об’єктів, пріоритетів і дій.
- Створення запитів до журналу systemd.
- Фільтрування даних журналу systemd за такими критеріями, як дата, служба або пріоритет.
- Налаштування постійного сховища журналу systemd і розміру журналу.
- Видалення старих даних журналу systemd.
- Отримання даних журналу systemd із системи відновлення або копії файлової системи.
- Розуміння взаємодії rsyslog із systemd-journald.
- Конфігурація logrotate.
- Знання syslog і syslog-ng.

Частковий список файлів, термінів та утиліт, що використовуються

- `/etc/rsyslog.conf`
- `/var/log/`
- `logger`

- `logrotate`
- `/etc/logrotate.conf`
- `/etc/logrotate.d/`
- `journalctl`
- `systemd-cat`
- `/etc/systemd/journald.conf`
- `/var/log/journal/`

108.2 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	108 Основні системні служби
Тема:	108.2 Системне журналювання
Урок:	1 з 2

Вступ

Журнали можуть бути найкращим другом системного адміністратора. Журнали — це файли (зазвичай текстові), де в хронологічному порядку реєструються всі системні та мережеві події з моменту завантаження системи. Таким чином, спектр інформації, яку можна знайти в журналах, включає практично всі аспекти системи: невдалі спроби автентифікації, помилки програм і служб, хости, заблоковані міжмережним екраном тощо. Як ви можете собі уявити, журнали роблять життя системних адміністраторів значно легшим, коли йдеться про усунення несправностей, перевірку ресурсів, виявлення аномальної поведінки програм тощо.

У цьому уроці ми обговоримо одну з найпоширеніших засобів журналювання, які зараз є в дистрибутивах GNU/Linux: `rsyslog`. Ми вивчимо різні типи журналів, які існують, де вони зберігаються, яку інформацію містять і як цю інформацію можна отримати та відфільтрувати. Ми також обговоримо, як журнали можна зберігати на централізованих серверах у IP-мережах, ротацію журналів і кільцевий буфер ядра.

Системне журналювання

У той момент, коли ядро та різні процеси у вашій системі починають працювати та спілкуватися один з одним, багато інформації генерується у формі повідомлень, які, здебільшого, надсилаються до журналів.

Без журналювання пошук події, яка сталася на сервері, викликав би головний біль у системних адміністраторів, тому важливо мати стандартизований і централізований спосіб відстеження будь-яких системних подій. Журнали є визначальними та показовими, коли йдеться про усунення несправностей і безпеку, а також є надійними джерелами даних для розуміння системної статистики та прогнозування тенденцій.

Залишаючи осторонь `systemd-journald` (який ми обговоримо в наступному уроці), журналювання традиційно оброблялося трьома основними спеціальними службами: `syslog`, `syslog-ng` (syslog нового покоління) і `rsyslog` (“ракетно-швидкісна система обробки журналів”). `rsyslog` приніс важливі покращення (такі як підтримка RELP) і став найпопулярнішим вибором на сьогодні. Кожна з цих служб збирає повідомлення від інших служб і програм і зберігає їх у файлах журналу, зазвичай у папці `/var/log`. Однак деякі служби піклуються про власні журнали (наприклад, веб-сервер Apache HTTPD або система друку CUPS). Подібним чином ядро Linux використовує кільцевий буфер у пам’яті для зберігання повідомлень журналу.

NOTE `RELP` означає *Reliable Event Logging Protocol* (Протокол надійної реєстрації подій) і розширює функціональні можливості протоколу системного журналу для забезпечення надійної доставки повідомлень.

Оскільки `rsyslog` став *де факто* стандартним засобом журналювання в усіх основних дистрибутивах, ми зосередимося на ньому в цьому уроці. `rsyslog` використовує модель клієнт-сервер. Клієнт і сервер можуть знаходитися на одному хості або на різних машинах. Повідомлення надсилаються та отримуються в певному форматі та можуть зберігатися на централізованих серверах `rsyslog` у IP-мережах. Демон rsyslog — `rsyslogd` — працює разом із `klogd` (який керує повідомленнями ядра). У наступних розділах буде розглянуто `rsyslog` та його інфраструктуру журналювання.

NOTE Демон — це служба, яка працює у фоновому режимі. Зверніть увагу на останню літеру `d` в назвах демонів: `klogd` або `rsyslogd`.

Типи журналів

Оскільки журнали є *змінними* даними, вони зазвичай знаходяться в `/var/log`. Грубо кажучи, їх можна класифікувати на *системні журнали* та *службові або програмні*

журнали.

Давайте розглянемо деякі системні журнали та інформацію, яку вони зберігають:

`/var/log/auth.log`

Діяльність, пов'язана з процесами автентифікації: зареєстровані користувачі, інформація `sudo`, завдання cron, невдалі спроби входу тощо.

`/var/log/syslog`

Централізований файл для практично всіх журналів, отриманих `rsyslogd`. Оскільки він містить дуже багато інформації, журнали розподіляються між іншими файлами відповідно до конфігурації, наданої в `/etc/rsyslog.conf`.

`/var/log/debug`

Налагоджувальна інформація з програм.

`/var/log/kern.log`

Повідомлення ядра.

`/var/log/messages`

Інформативні повідомлення, які стосуються не ядра, а інших служб. Це також місце призначення журналу віддаленого клієнта за замовчуванням у реалізації централізованого сервера журналу.

`/var/log/daemon.log`

Інформація, пов'язана з демонами або службами, що працюють у фоновому режимі.

`/var/log/mail.log`

Інформація, пов'язана з сервером електронної пошти, наприклад, postfix.

`/var/log/Xorg.0.log`

Інформація про графічну карту.

`/var/run/utmp` і **`/var/log/wtmp`**

Успішні входи.

`/var/log/btmp`

Невдалі спроби входу, наприклад, атака грубої сили через ssh.

`/var/log/faillog`

Невдалі спроби автентифікації.

`/var/log/lastlog`

Дата й час останніх входів користувачів.

Тепер розглянемо кілька прикладів журналів служб:

`/var/log/cups/`

Каталог для журналів *Загальної системи друку Unix*. Він зазвичай включає такі файли журналу за замовчуванням: `error_log`, `page_log` і `access_log`.

`/var/log/apache2/` або `/var/log/httpd`

Каталог для журналів *веб-сервера Apache*. Він зазвичай містить такі файли журналу за замовчуванням: `access.log`, `error_log` і `other_vhosts_access.log`.

`/var/log/mysql`

Каталог для журналів *системи керування реляційними базами даних MySQL*. Зазвичай він включає такі файли журналу за замовчуванням: `error_log`, `mysql.log` і `mysql-slow.log`.

`/var/log/samba/`

Каталог для журналів протоколу *Session Message Block* (SMB). Він зазвичай включає такі файли журналу за замовчуванням: `log.`, `log.nmbd` і `log.smbd`.

NOTE

Точна назва та вміст файлів журналу можуть відрізнятися в різних дистрибутивах Linux. Існують також журнали, присвячені певним дистрибутивам, наприклад `/var/log/dpkg.log` (містить інформацію про пакети `dpkg`) у Debian GNU/Linux та його похідних.

Читання журналів

Щоб читати файли журналу, спочатку переконайтеся, що ви є користувачем root або маєте дозвіл на читання файлу. Ви можете використовувати різні утиліти, такі як:

`less` або `more`

Командт, які дозволяють переглядати та прокручувати по одній сторінці за раз:

```
root@debian:~# less /var/log/auth.log
Sep 12 18:47:56 debian sshd[441]: Received SIGHUP; restarting.
Sep 12 18:47:56 debian sshd[441]: Server listening on 0.0.0.0 port 22.
Sep 12 18:47:56 debian sshd[441]: Server listening on :: port 22.
Sep 12 18:47:56 debian sshd[441]: Received SIGHUP; restarting.
Sep 12 18:47:56 debian sshd[441]: Server listening on 0.0.0.0 port 22.
```

```
Sep 12 18:47:56 debian sshd[441]: Server listening on :: port 22.
Sep 12 18:49:46 debian sshd[905]: Accepted password for carol from 192.168.1.65 port
44296 ssh2
Sep 12 18:49:46 debian sshd[905]: pam_unix(sshd:session): session opened for user carol
by (uid=0)
Sep 12 18:49:46 debian systemd-logind[331]: New session 2 of user carol.
Sep 12 18:49:46 debian systemd: pam_unix(systemd-user:session): session opened for user
carol by (uid=0)
(...)
```

`zless` або `zmore`

Те саме, що `less` і `more`, але використовується для журналів, які стискаються за допомогою `gzip` (загальна функція `logrotate`):

```
root@debian:~# zless /var/log/auth.log.3.gz
Aug 19 20:05:57 debian sudo:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/sbin/shutdown -h now
Aug 19 20:05:57 debian sudo: pam_unix(sudo:session): session opened for user root by
carol(uid=0)
Aug 19 20:05:57 debian lightdm: pam_unix(lightdm-greeter:session): session closed for
user lightdm
Aug 19 23:50:49 debian systemd-logind[333]: Watching system buttons on /dev/input/event2
(Power Button)
Aug 19 23:50:49 debian systemd-logind[333]: Watching system buttons on /dev/input/event3
(Sleep Button)
Aug 19 23:50:49 debian systemd-logind[333]: Watching system buttons on /dev/input/event4
(Video Bus)
Aug 19 23:50:49 debian systemd-logind[333]: New seat seat0.
Aug 19 23:50:49 debian sshd[409]: Server listening on 0.0.0.0 port 22.
(...)
```

`tail`

Перегляд останніх рядків у файлі (за замовчуванням 10 рядків). Потужність tail — значною мірою — полягає в перемикачі `-f`, який динамічно показуватиме нові рядки в міру їх додавання:

```
root@suse-server:~# tail -f /var/log/messages
2019-09-14T13:57:28.962780+02:00 suse-server sudo: pam_unix(sudo:session): session closed
for user root
2019-09-14T13:57:38.038298+02:00 suse-server sudo:    carol : TTY=pts/0 ; PWD=/home/carol
; USER=root ; COMMAND=/usr/bin/tail -f /var/log/messages
```

```
2019-09-14T13:57:38.039927+02:00 suse-server sudo: pam_unix(sudo:session): session opened
for user root by carol(uid=0)
2019-09-14T14:07:22+02:00 debian carol: appending new message from client to remote
server...
```

`head`

Переглянути перші рядки у файлі (за замовчуванням 10 рядків):

```
root@suse-server:~# head -5 /var/log/mail
2019-06-29T11:47:59.219806+02:00 suse-server postfix/postfix-script[1732]: the Postfix
mail system is not running
2019-06-29T11:48:01.355361+02:00 suse-server postfix/postfix-script[1925]: starting the
Postfix mail system
2019-06-29T11:48:01.391128+02:00 suse-server postfix/master[1930]: daemon started --
version 3.3.1, configuration /etc/postfix
2019-06-29T11:55:39.247462+02:00 suse-server postfix/postfix-script[3364]: stopping the
Postfix mail system
2019-06-29T11:55:39.249375+02:00 suse-server postfix/master[1930]: terminating on signal
15
```

`grep`

Утиліта фільтрації, яка дозволяє шукати певні рядки:

```
root@debian:~# grep "dhclient" /var/log/syslog
Sep 13 11:58:48 debian dhclient[448]: DHCPREQUEST of 192.168.1.4 on enp0s3 to 192.168.1.1
port 67
Sep 13 11:58:49 debian dhclient[448]: DHCPACK of 192.168.1.4 from 192.168.1.1
Sep 13 11:58:49 debian dhclient[448]: bound to 192.168.1.4 -- renewal in 1368 seconds.
(...)
```

Як ви могли помітити, вихідні дані друкуються в такому форматі:

- Мітка часу.
- Ім’я хосту, з якого надійшло повідомлення.
- Назва програми/служби, яка створила повідомлення.
- PID програми, яка створила повідомлення.
- Опис дії, що відбулася.

Є кілька прикладів, коли журнали є не текстовими, а двійковими файлами, і, отже, ви

повинні використовувати спеціальні команди для їх аналізу:

`/var/log/wtmp`

Використовуйте `who` (або `w`):

```
root@debian:~# who
root    pts/0        2020-09-14 13:05 (192.168.1.75)
root    pts/1        2020-09-14 13:43 (192.168.1.75)
```

`/var/log/btmp`

Використовуйте `utmpdump` або `last -f`:

```
root@debian:~# utmpdump /var/log/btmp
Utmp dump of /var/log/btmp
[6] [01287] [    ] [dave     ] [ssh:notty   ] [192.168.1.75        ] [192.168.1.75   ]
[2019-09-07T19:33:32,000000+0000]
```

`/var/log/faillog`

Використовуйте `faillog`:

```
root@debian:~# faillog -a | less
Login       Failures Maximum Latest                   On

root            0        0   01/01/70 01:00:00 +0100
daemon          0        0   01/01/70 01:00:00 +0100
bin             0        0   01/01/70 01:00:00 +0100
sys             0        0   01/01/70 01:00:00 +0100
sync            0        0   01/01/70 01:00:00 +0100
games           0        0   01/01/70 01:00:00 +0100
man             0        0   01/01/70 01:00:00 +0100
lp              0        0   01/01/70 01:00:00 +0100
mail            0        0   01/01/70 01:00:00 +0100
(...)
```

`/var/log/lastlog`

Використовуйте `lastlog`:

```
root@debian:~# lastlog | less
Username         Port     From             Latest
```

```
root                                      Never logged in
daemon                                    Never logged in
bin                                       Never logged in
sys                                       Never logged in
(...)
sync                                      Never logged in
avahi                                     Never logged in
colord                                    Never logged in
saned                                     Never logged in
hplip                                     Never logged in
carol            pts/1    192.168.1.75     Sat Sep 14 13:43:06 +0200 2019
dave             pts/3    192.168.1.75     Mon Sep  2 14:22:08 +0200 2019
```

NOTE Існують також графічні засоби для читання файлів журналів, наприклад: `gnome-logs` і `KSystemLog`.

Як повідомлення перетворюються на журнали

Наступний процес ілюструє, як повідомлення записується до файлу журналу:

1. Програми, служби та ядро записують повідомлення в спеціальні файли (сокети та буфери пам'яті), наприклад, `/dev/log` або `/dev/kmsg`.
2. `rsyslogd` отримує інформацію з сокетів або буферів пам'яті.
3. Залежно від правил, знайдених у `/etc/rsyslog.conf` та/або файлах `/etc/ryslog.d/`, `rsyslogd` переміщує інформацію до відповідного файлу журналу (зазвичай знаходиться в `/var/log` `).

NOTE Сокет — це спеціальний файл, який використовується для передачі інформації між різними процесами. Щоб отримати список усіх сокетів у вашій системі, ви можете використати команду `systemctl list-sockets --all`.

Об'єкти, пріоритети та дії

Конфігураційний файл `rsyslog` — це `/etc/rsylog.conf` (у деяких дистрибутивах ви також можете знайти файли конфігурації в `/etc/rsyslog.d/`). Зазвичай він розділений на три розділи: `MODULES`, `GLOBAL DIRECTIVES` і `RULES`. Давайте подивимося на них, вивчивши файл `rsyslog.conf` на нашому хості Debian GNU/Linux 10 (buster) — для цього можна використати `sudo less /etc/rsyslog.conf`.

`MODULES` містить підтримку модулів для журналювання, можливості повідомлень і

отримання журналу UDP/TCP:

```
#################
#### MODULES ####
#################

module(load="imuxsock") # provides support for local system logging
module(load="imklog")   # provides kernel logging support
#module(load="immark")  # provides --MARK-- message capability

# provides UDP syslog reception
#module(load="imudp")
#input(type="imudp" port="514")

# provides TCP syslog reception
#module(load="imtcp")
#input(type="imtcp" port="514")
```

`GLOBAL DIRECTIVES` дозволяють нам налаштувати низку речей, таких як журнали та дозволи каталогу журналів:

```
###########################
#### GLOBAL DIRECTIVES ####
###########################

#
# Use traditional timestamp format.
# To enable high precision timestamps, comment out the following line.
#
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

#
# Set the default permissions for all log files.
#
$FileOwner root
$FileGroup adm
$FileCreateMode 0640
$DirCreateMode 0755
$Umask 0022

#
# Where to place spool and state files
#
```

```
$WorkDirectory /var/spool/rsyslog

#
# Include all config files in /etc/rsyslog.d/
#
$IncludeConfig /etc/rsyslog.d/*.conf
```

В `RULES` визначені *facilities*, *priorities* і *actions*. Параметри в цьому розділі вказують демону журналювання фільтрувати повідомлення відповідно до певних правил і реєструвати їх або надсилати, куди потрібно. Щоб зрозуміти ці правила, ми повинні спочатку пояснити поняття засобів і пріоритетів `rsyslog`. Кожне повідомлення журналу отримує номер *facility* і ключове слово, пов'язане з внутрішньою підсистемою Linux, яка створює повідомлення:

Номер	**Ключове слово**	**Опис**
`0`	`kern`	Повідомлення ядра Linux
`1`	`користувач`	Повідомлення на рівні користувача
`2`	`пошта`	Поштова система
`3`	`демон`	Системні демони
`4`	`auth`, `authpriv`	Повідомлення безпеки/авторизації
`5`	`syslog`	Повідомлення syslogd
`6`	`lpr`	Підсистема рядкового принтера
`7`	`новини`	Підсистема новин мережі
`8`	`uucp`	Підсистема UUCP (протокол копіювання Unix-to-Unix).
`9`	`cron`	Демон годинника
`10`	`auth`, `authpriv`	Повідомлення безпеки/авторизації
`11`	`ftp`	Демон FTP (протокол передачі файлів).
`12`	`ntp`	Демон NTP (протокол мережевого часу).
`13`	`безпека`	Журнал аудиту

Номер	Ключове слово	Опис
14	консоль	Сповіщення журналу
15	cron	Демон годинника
16 - 23	від local0 до local7	Локальне використання 0 - 7

Крім того, кожному повідомленню присвоюється рівень *пріоритету*:

Код	Вага	Ключове слово	Опис
0	Emergency	emerg, panic	Система непридатна для використання
1	Alert	alert	Треба негайно діяти
2	Critical	crit	Критичні стани
3	Error	err, error	Аварійні стани
4	Warning	warn, warning	Стан попередження
5	Notice	notice	Нормальний, але важливий стан
6	Informational	info	Інформаційні повідомлення
7	Debug	debug	Повідомлення рівня налагодження

Ось фрагмент `rsyslog.conf` з нашої системи Debian GNU/Linux 10 (buster), який містить кілька прикладів правил:

```
###############
#### RULES ####
###############

# First some standard log files.  Log by facility.
#
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
```

```
user.*                          -/var/log/user.log

#
# Logging for the mail system.  Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info                       -/var/log/mail.info
mail.warn                       -/var/log/mail.warn
mail.err                        /var/log/mail.err

#
# Some "catch-all" log files.
#
*.=debug;\
        auth,authpriv.none;\
    news.none;mail.none     -/var/log/debug
*.=info;*.=notice;*.=warn;\
    auth,authpriv.none;\
    cron,daemon.none;\
    mail,news.none          -/var/log/messages
```

Формат правила такий: `<facility>.<priority>` `<action>`

Селектор `<facility>.<priority>` фільтрує повідомлення на відповідність. Рівні пріоритету є ієрархічно інклюзивними, що означає, що rsyslog відповідатиме повідомленням із вказаним пріоритетом і вищим за нього. `<action>` показує, яку дію виконати (куди надсилати повідомлення журналу). Ось кілька прикладів для наочності:

```
auth,authpriv.*                 /var/log/auth.log
```

Незалежно від їх пріоритету (`*`), усі повідомлення від об'єктів `auth` або `authpriv` надсилатимуться до `/var/log/auth.log`.

```
*.*;auth,authpriv.none          -/var/log/syslog
```

Усі повідомлення — незалежно від їхнього пріоритету (`*`) — з усіх об'єктів (`*`) — відкидаючи повідомлення з `auth` або `authpriv` (звідси суфікс `.none`) — буде записано в `/var/log/syslog` (знак дефіс (`-`) перед шляхом запобігає надмірному запису на диск). Зверніть увагу на крапку з комою (`;`), щоб розділити селектор, і кому (`,`), щоб об'єднати два засоби в одному правилі (`auth,authpriv`).

```
mail.err                        /var/log/mail.err
```

Повідомлення від засобу `mail` з рівнем пріоритету `error` або вищим (`critical`, `alert` або `emergency`) надсилатимуться до `/var/log/mail.err`.

```
*.=debug;\
        auth,authpriv.none;\
    news.none;mail.none     -/var/log/debug
```

Повідомлення від усіх об'єктів із пріоритетом `debug` і жодним іншим (`=`) записуватимуться до `/var/log/debug`, за винятком будь-яких повідомлень, що надходять від `auth`, `authpriv`, `news` і засобів `mail` (зверніть увагу на синтаксис: `;\`).

Записи в системному журналі вручну: `logger`

Команда `logger` стане в нагоді для сценаріїв оболонки або для тестування. `logger` додаватиме будь-яке отримане повідомлення до `/var/log/syslog` (або до `/var/log/messages` під час входу на віддалений центральний сервер журналу, як ви побачите далі в цьому уроці):

```
carol@debian:~$ logger this comment goes into "/var/log/syslog"
```

Щоб надрукувати останній рядок у `/var/log/syslog`, використовуйте команду `tail` з опцією `-1`:

```
root@debian:~# tail -1 /var/log/syslog
Sep 17 17:55:33 debian carol: this comment goes into /var/log/syslog
```

`rsyslog` як центральний сервер журналу

Щоб пояснити цю тему, ми збираємося додати новий хост до наших налаштувань. Схема така:

Роль	Ім'я хоста	ОС	IP-адреса
Центральний сервер журналу	`suse-server`	openSUSE Leap 15.1	192.168.1.6

Роль	**Ім'я хоста**	**ОС**	**IP-адреса**
Клієнт	`debian`	Debian GNU/Linux 10 (buster)	192.168.1.4

Почнемо з налаштування сервера. Перш за все, ми переконаємося, що `rsyslog` запущений і працює:

```
root@suse-server:~# systemctl status rsyslog
 rsyslog.service - System Logging Service
   Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2019-09-17 18:45:58 CEST; 7min ago
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
 Main PID: 832 (rsyslogd)
    Tasks: 5 (limit: 4915)
   CGroup: /system.slice/rsyslog.service
           └─832 /usr/sbin/rsyslogd -n -iNONE
```

openSUSE поставляється зі спеціальним файлом конфігурації для віддаленого журналювання: `/etc/rsyslog.d/remote.conf`. Увімкніть отримання повідомлень від клієнтів (віддалених хостів) через TCP. Ми повинні розкоментувати рядки, які завантажують модуль і запускають сервер TCP на порту 514:

```
# ######### Receiving Messages from Remote Hosts ##########
# TCP Syslog Server:
# provides TCP syslog reception and GSS-API (if compiled to support it)
$ModLoad imtcp.so  # load module
##$UDPServerAddress 10.10.0.1  # force to listen on this IP only
$InputTCPServerRun 514  # Starts a TCP server on selected port

# UDP Syslog Server:
#$ModLoad imudp.so  # provides UDP syslog reception
##$UDPServerAddress 10.10.0.1  # force to listen on this IP only
#$UDPServerRun 514  # start a UDP syslog server at standard port 514
```

Коли це буде зроблено, ми повинні перезапустити службу rsyslog і перевірити, чи сервер прослуховує порт 514:

```
root@suse-server:~# systemctl restart rsyslog
root@suse-server:~# netstat -nltp | grep 514
```

```
[sudo] password for root:
tcp        0      0 0.0.0.0:514             0.0.0.0:*               LISTEN
2263/rsyslogd
tcp6       0      0 :::514                  :::*                    LISTEN
2263/rsyslogd
```

Далі ми повинні відкрити порти в брандмауері та перезавантажити конфігурацію:

```
root@suse-server:~# firewall-cmd --permanent --add-port 514/tcp
success
root@suse-server:~# firewall-cmd --reload
success
```

NOTE З появою openSUSE Leap 15.0 `firewalld` повністю замінив класичний `SuSEFirewall2`.

Шаблони та умови фільтрів

За замовчуванням журнали клієнта будуть записані у файл `/var/log/messages` сервера разом із журналами самого сервера. Однак ми створимо *шаблон* і *умову фільтра*, щоб журнали нашого клієнта зберігалися у власних чітких каталогах. Для цього ми додамо наступне до `/etc/rsyslog.conf` (або `/etc/rsyslog.d/remote.conf`):

```
$template RemoteLogs,"/var/log/remotehosts/%HOSTNAME%/%$NOW%.%syslogseverity-text%.log"
if $FROMHOST-IP=='192.168.1.4' then ?RemoteLogs
& stop
```

Шаблон

Шаблон відповідає першому рядку та дозволяє вказати формат імен журналів за допомогою динамічної генерації імен файлів. Шаблон складається з:

- Директиви шаблону (`$template`)
- Назви шаблону (`RemoteLogs`)
- Тексту шаблону (`"/var/log/remotehosts/%HOSTNAME%/%$NOW%.%syslogseverity-text%.log"`)
- Параметрів (необов’язково)

Наш шаблон називається `RemoteLogs`, і його текст складається зі шляху в `/var/log`. Усі журнали нашого віддаленого хоста потраплять до каталогу `remotehosts`, де буде створено

підкаталог на основі імені хоста машини (`%HOSTNAME%`). Ім'я кожного файлу в цьому каталозі складатиметься з дати (`%$NOW%`), рівня серйозності (він же пріоритет) повідомлення в текстовому форматі (`%syslogseverity-text%`) і суфікса `.log`. Слова між знаками відсотка є *властивостями* і дозволяють отримати доступ до вмісту повідомлення журналу (дата, пріоритет тощо). Повідомлення `syslog` має низку добре визначених властивостей, які можна використовувати в шаблонах. Доступ до цих властивостей здійснюється — і їх можна змінювати — за допомогою так званого *замінника властивостей*, який передбачає запис їх між знаками відсотків.

Стан фільтра

Решта два рядки відповідають умові фільтра та пов'язаній з ним дії:

- Фільтр на основі виразів (`if $FROMHOST-IP=='192.168.1.4'`)
- Дія (`then ?RemoteLogs`, `& stop`)

Перший рядок перевіряє IP-адресу віддаленого хоста, який надсилає журнал, і, якщо вона збігається з адресою нашого клієнта Debian, застосовує шаблон `RemoteLogs`. Останній рядок (`& stop`) гарантує, що повідомлення не надсилаються одночасно до `/var/log/messages` (а лише до файлів у каталозі `/var/log/remotehosts`).

NOTE Щоб дізнатися більше про шаблони, властивості та правила, ви можете переглянути сторінку посібника для `rsyslog.conf`.

Після оновлення конфігурації ми знову запустимо `rsyslog` і підтвердимо, що в `/var/log` ще немає каталогу `remotehosts`:

```
root@suse-server:~# systemctl restart rsyslog
root@suse-server:~# ls /var/log/
acpid             chrony     localmessages   pbl.log          Xorg.0.log
alternatives.log  cups       mail            pk_backend_zypp  Xorg.0.log.old
apparmor          firebird   mail.err        samba            YaST2
audit             firewall   mail.info       snapper.log      zypp
boot.log          firewalld  mail.warn       tallylog         zypper.log
boot.msg          krb5       messages        tuned
boot.omsg         lastlog    mysql           warn
btmp              lightdm    NetworkManager  wtmp
```

Сервер налаштовано. Далі ми налаштуємо клієнт.

Знову ж таки, ми повинні переконатися, що `rsyslog` встановлено та працює:

```
root@debian:~# sudo systemctl status rsyslog
 rsyslog.service - System Logging Service
   Loaded: loaded (/lib/systemd/system/rsyslog.service; enabled; vendor preset:
   Active: active (running) since Thu 2019-09-17 18:47:54 CEST; 7min ago
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
 Main PID: 351 (rsyslogd)
    Tasks: 4 (limit: 4915)
   CGroup: /system.slice/rsyslog.service
           └─351 /usr/sbin/rsyslogd -n
```

У нашому прикладі середовища ми реалізували розпізнавання імен на клієнті, додавши рядок `192.168.1.6 suse-server` до `/etc/hosts`. Таким чином, ми можемо звертатися до сервера або за назвою (`suse-server`), або за IP-адресою (`192.168.1.6`).

Наш клієнт Debian не постачається з файлом `remote.conf` у `/etc/rsyslog.d/`, тому ми застосуємо наші налаштування у `/etc/rsyslog.conf`. У кінці файлу ми напишемо такий рядок:

```
*.* @@suse-server:514
```

Нарешті, ми перезапускаємо `rsyslog`.

```
root@debian:~# systemctl restart rsyslog
```

Тепер давайте повернемося до нашої машини `suse-server` і перевіримо існування `remotehost` в `/var/log`:

```
root@suse-server:~# ls /var/log/remotehosts/debian/
2019-09-17.info.log  2019-09-17.notice.log
```

У нас уже є два журнали всередині `/var/log/remotehosts`, як описано в нашому шаблоні. Щоб завершити цей розділ, ми запускаємо `tail -f 2019-09-17.notice.log` на `suse-server`, надсилаючи журнал *вручну* з нашого клієнта Debian і підтверджуючи, що повідомлення додаються до файлу журналу як очікується (параметр `-t` надає тег для нашого повідомлення):

```
root@suse-server:~# tail -f /var/log/remotehosts/debian/2019-09-17.notice.log
2019-09-17T20:57:42+02:00 debian dbus[323]: [system] Successfully activated service
```

```
'org.freedesktop.nm_dispatcher'
2019-09-17T21:01:41+02:00 debian anacron[1766]: Anacron 2.3 started on 2019-09-17
2019-09-17T21:01:41+02:00 debian anacron[1766]: Normal exit (0 jobs run)
```

```
carol@debian:~$ logger -t DEBIAN-CLIENT Hi from 192.168.1.4
```

```
root@suse-server:~# tail -f /var/log/remotehosts/debian/2019-09-17.notice.log
2019-09-17T20:57:42+02:00 debian dbus[323]: [system] Successfully activated service
'org.freedesktop.nm_dispatcher'
2019-09-17T21:01:41+02:00 debian anacron[1766]: Anacron 2.3 started on 2019-09-17
2019-09-17T21:01:41+02:00 debian anacron[1766]: Normal exit (0 jobs run)
2019-09-17T21:04:21+02:00 debian DEBIAN-CLIENT: Hi from 192.168.1.4
```

Механізм ротації журналів

Журнали регулярно чергуються, що служить двом основним цілям:

- Запобігання використанню старими файлами журналів дискового простору більше, ніж необхідно.
- Зберігання журналів до керованої довжини для зручності перегляду.

Утиліта, яка відповідає за ротацію (або циклічність) журналів, — це `logrotate`, і її робота включає такі дії, як переміщення файлів журналу під новою назвою, їх архівування та/або стиснення, іноді надсилання їх електронною поштою системному адміністратору та зрештою їх видалення, коли вони застарівають. Існують різні угоди щодо іменування цих змінених файлів журналу (наприклад, додавання суфікса з датою до назви файлу); однак просто додавання суфікса з цілим числом є звичайною практикою:

```
root@debian:~# ls /var/log/messages*
/var/log/messages  /var/log/messages.1  /var/log/messages.2.gz  /var/log/messages.3.gz
/var/log/messages.4.gz
```

Давайте тепер пояснимо, що станеться під час наступної ротації журналу:

1. `messages.4.gz` буде видалено та втрачено.
2. Вміст `messages.3.gz` буде переміщено до `messages.4.gz`.
3. Вміст `messages.2.gz` буде переміщено до `messages.3.gz`.
4. Вміст `messages.1` буде переміщено до `messages.2.gz`.

5. Вміст `messages` буде переміщено до `messages.1`, а `messages` буде порожнім і готовим до реєстрації нових записів журналу.

Зауважте, як, згідно з директивами `logrotate`, які ви незабаром побачите, три старіші файли журналу стискаються, а два найновіші – ні. Крім того, ми будемо зберігати журнали за останні 4-5 тижнів. Щоб прочитати повідомлення 1-тижневої давності, ми звернемося до `messages.1` (і так далі).

`logrotate` запускається як автоматизований процес або завдання cron щодня за допомогою сценарію `/etc/cron.daily/logrotate` і читає файл конфігурації `/etc/logrotate.conf`. Цей файл містить кілька глобальних параметрів і добре прокоментований, до кожного параметра введено коротке пояснення його призначення:

```
carol@debian:~$ sudo less /etc/logrotate.conf
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

(...)
```

Як бачите, файли конфігурації в `/etc/logrotate.d` для певних пакунків також включені. Ці файли здебільшого містять локальні визначення та вказують файли журналу для ротації (пам'ятайте, що локальні визначення мають пріоритет над глобальними, а пізніші визначення замінюють попередні). Далі наведено уривок визначення в `/etc/logrotate.d/rsyslog`:

```
/var/log/messages
{
        rotate 4
        weekly
```

```
        missingok
        notifempty
        compress
        delaycompress
        sharedscripts
        postrotate
                invoke-rc.d rsyslog rotate > /dev/null
        endscript
}
```

Як ви бачите, кожна директива відокремлена від свого значення пробілом та/або необов'язковим знаком рівності (=). Рядки між `postrotate` і `endscript` повинні з'являтися в рядках окремо. Пояснення полягає в наступному:

`rotate 4`

Зберігає журнали на 4 тижні.

`weekly`

Щотижня проводить ротацію файлів журналів.

`missingok`

Не видає повідомлення про помилку, якщо файл журналу відсутній; просто переходить до наступного.

`notifempty`

Не проводить ротацію журналу, якщо він порожній.

`compress`

Стискає файли журналу за допомогою `gzip` (за замовчуванням).

`delaycompress`

Відкладає стиснення попереднього файлу журналу до наступного циклу ротації (ефективно, лише коли використовуються в поєднанні зі стисненням). Це корисно, коли програмі не можна наказати закрити свій файл журналу, і, таким чином, вона може продовжувати запис у попередній файл журналу протягом деякого часу.

`sharedscripts`

Пов'язано зі сценаріями *prerotate* і *postrotate*. Щоб запобігти багаторазовому виконанню сценарію, запустіть сценарії лише один раз незалежно від того, скільки файлів журналу відповідає заданому шаблону (наприклад, `/var/log/mail/*`). Сценарії не запускатимуться, якщо жоден із журналів у шаблоні не вимагає ротації. Крім того, якщо

сценарії завершуються з помилками, будь-які інші дії не виконуватимуться для журналів.

`postrotate`

Вказує на початок сценарію *postrotate*.

`invoke-rc.d rsyslog rotate > /dev/null`

Використовуйте `/bin/sh`, щоб запустити `invoke-rc.d rsyslog rotate > /dev/null` після ротації журналів.

`endscript`

Вкажіть кінець сценарію *postrotate*.

NOTE Щоб отримати повний список директив і пояснень, зверніться до man-сторінки для `logrotate.conf`.

Кільцевий буфер ядра

Оскільки ядро генерує кілька повідомлень до того, як `rsyslogd` стане доступним під час завантаження, необхідний механізм реєстрації цих повідомлень. Ось тут і вступає в дію *кільцевий буфер ядра*. Це структура даних фіксованого розміру, і, отже, у міру збільшення нових повідомлень найстаріші зникатимуть.

Команда `dmesg` виводить кільцевий буфер ядра. Через розмір буфера цю команду зазвичай використовують у поєднанні з утилітою фільтрації тексту `grep`. Наприклад, щоб шукати повідомлення, пов'язані з пристроями універсальної послідовної шини:

```
root@debian:~# dmesg | grep "usb"
[    1.241182] usbcore: registered new interface driver usbfs
[    1.241188] usbcore: registered new interface driver hub
[    1.250968] usbcore: registered new device driver usb
[    1.339754] usb usb1: New USB device found, idVendor=1d6b, idProduct=0001, bcdDevice=
4.19
[    1.339756] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
(...)
```

Вправи до посібника

1. Які утиліти/команди ви б використали в наведених нижче сценаріях:

Призначення та файл журналу	**Утиліта**
Читання `/var/log/syslog.7.gz`	
Читання `/var/log/syslog`	
Фільтр за словом `renewal` в `/var/log/syslog`	
Читання `/var/log/faillog`	
Динамічне читання `/var/log/syslog`	

2. Переставте наступні записи журналу таким чином, щоб вони представляли дійсне повідомлення журналу з належною структурою:
   - `debian-server`
   - `sshd`
   - `[515]:`
   - `Sep 13 21:47:56`
   - `Server listening on 0.0.0.0 port 22`

   Правильний порядок:

3. Які правила ви б додали до `/etc/rsyslog.conf`, щоб виконати кожну з наступних дій:
   - Надсилати всі повідомлення засобом `mail` і з пріоритететом/серйозністю `crit` (і вище) до `/var/log/mail.crit`:
   - Надсилати всі повідомлення засобом `mail` з пріоритетами `alert` і `emergency` до `/var/log/mail.urgent`:
   - За винятком тих, що надходять із засобів `cron` і `ntp`, надсилати всі повідомлення — незалежно від їхнього засобу та пріоритету — до `/var/log/allmessages`:

- Спершу правильно налаштуйте всі необхідні параметри, надішліть усі повідомлення засобом `mail` на віддалений хост, IP-адреса якого `192.168.1.88`, використовуючи TCP і вказавши порт за замовчуванням:

- Незалежно від об'єкту, надсилайте всі повідомлення з пріоритетом `warning` (_тільки з пріоритетом `warning` _) до `/var/log/warnings`, запобігаючи надмірному запису на диск:

4. Розгляньте наступні дані з `/etc/logrotate.d/samba` та поясніть різні варіанти:

```
carol@debian:~$ sudo head -n 11 /etc/logrotate.d/samba
/var/log/samba/log.smbd {
        weekly
        missingok
        rotate 7
        postrotate
                [ ! -f /var/run/samba/smbd.pid ] || /etc/init.d/smbd reload > /dev/null
        endscript
        compress
        delaycompress
        notifempty
}
```

Варіант	**Значення**
`weekly`	
`missingok`	
`rotate 7`	
`postrotate`	
`endscript`	
`compress`	
`delaycompress`	
`notifyempty`	

Дослідницькі вправи

1. У розділі “Шаблони та умови фільтрації” ми використовували *фільтр на основі виразу* як умову фільтрації. *Фільтри на основі властивостей* є іншим типом фільтрів, унікальних для `rsyslogd`. Переробіть наш *фільтр на основі виразів* у *фільтр на основі властивостей*:

Фільтр на основі виразів	**Фільтр на основі властивостей**
`if $FROMHOST-IP=='192.168.1.4' then ?RemoteLogs`	

2. `omusrmsg` — це вбудований модуль `rsyslog`, який полегшує сповіщення користувачів (він надсилає повідомлення журналу на термінал користувача). Напишіть правило для надсилання всіх *надзвичайних* повідомлень з усіх об’єктів як `root`, так і звичайному користувачеві `carol`.

Підсумки

На цьому уроці ви вивчили:

- Ведення журналу має вирішальне значення для адміністрування системи.
- `rsyslogd` - це утиліта, яка відповідає за збереження акуратності та порядку журналів.
- Деякі служби піклуються про власні журнали.
- Грубо кажучи, журнали можна класифікувати на системні журнали та журнали служб/програм.
- Існує ряд утиліт, зручних для читання журналів: `less`, `more`, `zless`, `zmore`, `grep`, `head` і `tail`.
- Більшість файлів журналу є звичайними текстовими файлами; однак існує невелика кількість двійкових файлів журналу.
- Що стосується журналу, `rsyslogd` отримує відповідну інформацію зі спеціальних файлів (сокетів і буферів пам'яті) перед її обробкою.
- Для класифікації журналів `rsyslogd` використовує правила в `/etc/rsyslog.conf` або `/etc/rsyslog.d/*`.
- Будь-який користувач може вручну вводити власні повідомлення в системний журнал за допомогою утиліти `logger`.
- `rsyslog` дозволяє зберігати всі журнали в IP-мережах на централізованому сервері журналів.
- Шаблони стають у пригоді для динамічного форматування імен файлів журналу.
- Ротація журналів має подвійну мету: запобігти використанню надмірного дискового простору старими журналами та щоб зробити консультаційні журнали керованими.

Відповіді до вправ посібника

1. Які утиліти/команди ви б використали в наведених нижче сценаріях:

Призначення та файл журналу	**Утиліта**
Читання `/var/log/syslog.7.gz`	`zmore` або `zless`
Читання `/var/log/syslog`	`more` або `less`
Фільтр за словом `renewal` в `/var/log/syslog`	`grep`
Читання `/var/log/faillog`	`faillog -a`
Динамічне читання `/var/log/syslog`	`tail -f`

2. Переставте наступні записи журналу таким чином, щоб вони представляли дійсне повідомлення журналу з належною структурою:
   - `debian-server`
   - `sshd`
   - `[515]:`
   - `Sep 13 21:47:56`
   - `Server listening on 0.0.0.0 port 22`

   Правильний порядок:

   ```
   Sep 13 21:47:56 debian-server sshd[515]: Server listening on 0.0.0.0 port 22
   ```

3. Які правила ви б додали до `/etc/rsyslog.conf`, щоб виконати кожну з наступних дій:
   - Надсилати всі повідомлення засобом `mail` і з пріоритететом/серйозністю `crit` (і вище) до `/var/log/mail.crit`:

     ```
     mail.crit                /var/log/mail.crit
     ```

   - Надсилати всі повідомлення засобом `mail` з пріоритетами `alert` і `emergency` до `/var/log/mail.urgent`:

     ```
     mail.alert                       /var/log/mail.urgent
     ```

- За винятком тих, що надходять із засобів `cron` і `ntp`, надсилати всі повідомлення — незалежно від їхнього засобу та пріоритету — до `/var/log/allmessages`:

```
*.*;cron.none;ntp.none                  /var/log/allmessages
```

- Спершу правильно налаштуйте всі необхідні параметри, надішліть усі повідомлення засобом `mail` на віддалений хост, IP-адреса якого `192.168.1.88`, використовуючи TCP і вказавши порт за замовчуванням:

```
mail.* @@192.168.1.88:514
```

- Незалежно від об'єкту, надсилайте всі повідомлення з пріоритетом `warning` (_тільки з пріоритетом `warning` _) до `/var/log/warnings`, запобігаючи надмірному запису на диск:

```
*.=warning                          -/var/log/warnings
```

4. Розгляньте наступні дані з `/etc/logrotate.d/samba` та поясніть різні варіанти:

```
carol@debian:~$ sudo head -n 11 /etc/logrotate.d/samba
/var/log/samba/log.smbd {
        weekly
        missingok
        rotate 7
        postrotate
                [ ! -f /var/run/samba/smbd.pid ] || /etc/init.d/smbd reload > /dev/null
        endscript
        compress
        delaycompress
        notifempty
}
```

Варіант	**Значення**
`weekly`	Щотижнева ротація файлів журналу.
`missingok`	Не видавати повідомлення про помилку, якщо журнал відсутній; просто перейти до наступного.

Варіант	**Значення**
`rotate 7`	Зберігання 7 тижнів інформації про невиконанні завдання.
`postrotate`	Запуск сценарію у наступному рядку після ротації журналів.
`endscript`	Визначення кінця сценарію *postrotate*.
`compress`	Стиснення журналів за допомогою `gzip`.
`delaycompress`	У поєднанні з `compress` відкладення стиснення до наступного циклу ротації.
`notifyempty`	Не проводити ротацію журнала, якщо він порожній.

Відповіді до дослідницьких вправ

1. У розділі “Шаблони та умови фільтрації” ми використовували *фільтр на основі виразу* як умову фільтрації. *Фільтри на основі властивостей* є іншим типом фільтрів, унікальних для `rsyslogd`. Переробіть наш *фільтр на основі виразів* у *фільтр на основі властивостей*:

Фільтр на основі виразів	**Фільтр на основі властивостей**
`if $FROMHOST-IP=='192.168.1.4' then ?RemoteLogs`	`:fromhost-ip, isequal, "192.168.1.4" ?RemoteLogs`

2. `omusrmsg` — це вбудований модуль `rsyslog`, який полегшує сповіщення користувачів (він надсилає повідомлення журналу на термінал користувача). Напишіть правило для надсилання всіх *надзвичайних* повідомлень з усіх об’єктів як `root`, так і звичайному користувачеві `carol`.

```
*.emerg                         :omusrmsg:root,carol
```

Linux Professional Institute

108.2 Урок 2

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	108 Основні системні служби
Тема:	108.2 Системне журналювання
Урок:	2 з 2

Вступ

Із загальним прийняттям `systemd` усіма основними дистрибутивами, демон журналу (`systemd-journald`) став стандартною службою журналювання. У цьому уроці ми обговоримо, як він працює та як ви можете використовувати його, щоб робити ряд речей: запитувати дані з нього, фільтрувати його інформацію за різними критеріями, налаштовувати його зберігання та розмір, видаляти старі дані, отримувати їх із системи відновлення або копії файлової системи та — останнє, але не менш важливе — зрозуміти його взаємодію з `rsyslogd`.

Основи `systemd`

Вперше представлений у Fedora, `systemd` поступово замінив SysV Init як *de facto* системний і сервісний менеджер у більшості основних дистрибутивах Linux. Серед його сильних сторін можна виділити наступне:

- Простота конфігурації: unit-файли на відміну від сценаріїв ініціалізації SysV.
- Універсальне керування: окрім демонов і процесів, він також керує пристроями,

сокетами та точками монтування.

- Зворотна сумісність із SysV Init і Upstart.
- Паралельне завантаження під час завантаження ОС: служби завантажуються паралельно на відміну від того, як Sysv Init завантажує їх послідовно.
- Він містить службу журналювання під назвою *journal*, яка має такі переваги:
  - Централізує всі журнали в одному місці.
  - Не потребує ротації журналів.
  - Журнали можна вимкнути, завантажити в оперативну пам'ять або зробити постійними.

Модулі та цілі

`systemd` працює з *модулями*. Модулем є будь-який ресурс, яким може керувати `systemd` (наприклад, мережа, bluetooth тощо). Модулі, у свою чергу, керуються *файлами модулів*. Це звичайні текстові файли, які містяться в `/lib/systemd/system` і містять параметри конфігурації — у формі *розділів* і *директив* — для певного ресурсу, яким потрібно керувати. Існує декілька типів модулів: `service`, `mount`, `automount`, `swap`, `timer`, `device`, `socket`, `path`, `timer`, `snapshot`, `slice`, `scope` і `target`. Таким чином, кожне ім'я файлу блоку відповідає шаблону `<назва_ресурсу>.<тип_модуля>` (наприклад, `reboot.service`).

Target — це особливий тип модулю, який нагадує класичні рівні запуску SysV Init. Це пояснюється тим, що *target-модуль* об'єднує різні ресурси для представлення певного стану системи (наприклад, `graphical.target` подібний до `runlevel 5` тощо). Щоб перевірити поточну ціль у вашій системі, скористайтеся командою `systemctl get-default`:

```
carol@debian:~$ systemctl get-default
graphical.target
```

З іншого боку, цілі та рівні виконання відрізняються тим, що перші взаємопов'язані, а другі – ні. Таким чином, ціль може викликати інші цілі, що неможливо з рівнями запуску.

NOTE Пояснення того, як працюють модулі `systemd`, виходить за рамки цього уроку.

Системний журнал: `systemd-journald`

`systemd-journald` — це системна служба, яка піклується про отримання журнальної інформації з різноманітних джерел: повідомлень ядра, простих і структурованих

системних повідомлень, стандартного виведення та стандартних помилок служб, а також записів аудиту від підсистеми аудиту ядра (для додаткових відомостей дивіться man-сторінку для `systemd-journald`). Його місія полягає у створенні та підтримці структурованого та індексованого журналу.

Його конфігураційний файл має назву `/etc/systemd/journald.conf` і, як і з будь-якою іншою службою, ви можете використати команду `systemctl`, щоб *запустити* його, *перезапустити*, *зупинити* або — просто — перевірити його *статус* :

```
root@debian:~# systemctl status systemd-journald
 systemd-journald.service - Journal Service
   Loaded: loaded (/lib/systemd/system/systemd-journald.service; static; vendor preset:
enabled)
   Active: active (running) since Sat 2019-10-12 13:43:06 CEST; 5min ago
     Docs: man:systemd-journald.service(8)
           man:journald.conf(5)
 Main PID: 178 (systemd-journal)
   Status: "Processing requests..."
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/systemd-journald.service
           └─178 /lib/systemd/systemd-journald
(...)
```

Також можливі файли конфігурації типу `journal.conf.d/*.conf`, які можуть включати конфігурації, що стосуються окремих пакунків (щоб дізнатися більше, зверніться до man-сторінки `journald.conf`).

Якщо журнал ввімкнено, він може зберігатися або постійно на диску, або в непостійній формі у файловій системі на основі оперативної пам'яті. Журнал не є звичайним текстовим файлом, це двійковий файл. Таким чином, ви не можете використовувати інструменти аналізу тексту, такі як `less` або `more`, щоб прочитати його вміст; замість них використовується команда `journalctl`.

Запит щодо вмісту журналу

`journalctl` — це утиліта, яку ви використовуєте для запиту до журналу `systemd`. Щоб відкрити його, ви повинні або бути адміністратором, або використовувати `sudo`. Якщо надіслати запит без параметрів, він роздрукує весь журнал у хронологічному порядку (першими у списку будуть найстаріші записи):

```
root@debian:~# journalctl
```

```
-- Logs begin at Sat 2019-10-12 13:43:06 CEST, end at Sat 2019-10-12 14:19:46 CEST. --
Oct 12 13:43:06 debian kernel: Linux version 4.9.0-9-amd64 (debian-kernel@lists.debian.org)
(...)
Oct 12 13:43:06 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID=b6be6117-5226-4a8a-bade-2db35ccf4cf4 ro qu
(...)
```

Ви можете робити більш конкретні запити, використовуючи кілька перемикачів:

`-r`

Повідомлення журналу будуть надруковані у зворотному порядку:

```
root@debian:~# journalctl -r
-- Logs begin at Sat 2019-10-12 13:43:06 CEST, end at Sat 2019-10-12 14:30:30 CEST. --
Oct 12 14:30:30 debian sudo[1356]: pam_unix(sudo:session): session opened for user root
by carol(uid=0)
Oct 12 14:30:30 debian sudo[1356]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -r
Oct 12 14:19:53 debian sudo[1348]: pam_unix(sudo:session): session closed for user root
(...)
```

`-f`

Він друкуватиме останні повідомлення журналу та продовжуватиме друкувати нові записи, щойно вони будуть додані до журналу – подібно до `tail -f`:

```
root@debian:~# journalctl -f
-- Logs begin at Sat 2019-10-12 13:43:06 CEST. --
(...)
Oct 12 14:44:42 debian sudo[1356]: pam_unix(sudo:session): session closed for user root
Oct 12 14:44:44 debian sudo[1375]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -f
Oct 12 14:44:44 debian sudo[1375]: pam_unix(sudo:session): session opened for user root
by carol(uid=0)

(...)
```

`-e`

Він перейде в кінець журналу, щоб останні записи було видно в межах сторінки:

```
root@debian:~# journalctl -e
```

```
(...)
Oct 12 14:44:44 debian sudo[1375]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -f
Oct 12 14:44:44 debian sudo[1375]: pam_unix(sudo:session): session opened for user root
by carol(uid=0)
Oct 12 14:45:57 debian sudo[1375]: pam_unix(sudo:session): session closed for user root
Oct 12 14:48:39 debian sudo[1378]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -e
Oct 12 14:48:39 debian sudo[1378]: pam_unix(sudo:session): session opened for user root
by carol(uid=0)
```

`-n <value>, --lines=<value>`

Він надрукує останні рядки в кількості *value* (якщо `<value>` не вказано, за замовчуванням воно дорівнює 10):

```
root@debian:~# journalctl -n 5
(...)
Oct 12 14:44:44 debian sudo[1375]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -f
Oct 12 14:44:44 debian sudo[1375]: pam_unix(sudo:session): session opened for user root
by carol(uid=0)
Oct 12 14:45:57 debian sudo[1375]: pam_unix(sudo:session): session closed for user root
Oct 12 14:48:39 debian sudo[1378]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root ;
COMMAND=/bin/journalctl -e
Oct 12 14:48:39 debian sudo[1378]: pam_unix(sudo:session): session opened for user root
by carol(uid=0)
```

`-k`, `--dmesg`

Еквівалент використання команди `dmesg`:

```
root@debian:~# journalctl -k
-- Logs begin at Sat 2019-10-12 13:43:06 CEST, end at Sat 2019-10-12 14:53:20 CEST. --
Oct 12 13:43:06 debian kernel: Linux version 4.9.0-9-amd64 (debian-
kernel@lists.debian.org) (gcc version 6.3.0 20170516 (Debian 6.3.0-18
Oct 12 13:43:06 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID=b6be6117-5226-4a8a-bade-2db35ccf4cf4 ro qu
Oct 12 13:43:06 debian kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating
point registers'
Oct 12 13:43:06 debian kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
(...)
```

Навігація та пошук у журналі

Ви можете переглядати вихідні дані журналу за допомогою:

- PageUp, PageDown і клавіш зі стрілками для переміщення вгору, вниз, вліво і вправо.
- `>`, щоб перейти до кінця виведення.
- `<` для переходу на початок виведення.

Ви можете шукати рядки як вперед, так і назад від поточної позиції перегляду:

- Пошук вперед: натисніть `/` і введіть рядок для пошуку, потім натисніть Enter.
- Пошук у зворотному напрямку: натисніть `?` і введіть рядок для пошуку, потім натисніть Enter.

Щоб переходити між збігами в пошуках, використовуйте `N`, щоб перейти до наступного збігу, і `Shift` + `N`, щоб перейти до попереднього.

Фільтрування даних журналу

Журнал дозволяє фільтрувати дані журналу за різними критеріями:

Номер завантаження

`--list-boots`

У ньому перераховані всі доступні завантаження. Вихідні дані складаються з трьох колонок; перший визначає номер завантаження (`0` стосується поточного завантаження, `-1` — попереднього, `-2` передує попередньому і так далі); другий стовпець — ідентифікатор завантаження; третій показує мітки часу:

```
root@debian:~# journalctl --list-boots
 0 83df3e8653474ea5aed19b41cdb45b78 Sat 2019-10-12 18:55:41 CEST—Sat 2019-10-12
19:02:24 CEST
```

`-b`, `--boot`

Він показує всі повідомлення від поточного завантаження. Щоб переглянути повідомлення журналу з попередніх завантажень, просто додайте параметр offset, як описано вище. Наприклад, щоб надрукувати повідомлення з попереднього завантаження, ви введете `journalctl -b -1`. Однак пам'ятайте, що для відновлення інформації з попередніх журналів необхідно ввімкнути постійність журналу (ви дізнаєтеся, як це зробити в наступному розділі):

```
root@debian:~# journalctl -b -1
Specifying boot ID has no effect, no persistent journal was found
```

Пріоритет

`-p`

Цікаво, що ви також можете фільтрувати за серйозністю/пріоритетом за допомогою параметра `-p`:

```
root@debian:~# journalctl -b -0 -p err
-- No entries --
```

Журнал повідомляє, що поки що не було жодних повідомлень із пріоритетом `error` (або вище) від поточного завантаження. Примітка: `-b -0` можна опустити, коли йдеться про поточне завантаження.

NOTE Зверніться до попереднього уроку, щоб отримати повний список усіх рівнів серйозності `syslog` (так званих пріоритетів).

Проміжок часу

Ви можете наказати `journalctl` друкувати лише повідомлення, зареєстровані протягом певного періоду часу, використовуючи перемикачі `--since` і `--until`. Специфікація дати має відповідати формату `YYYY-MM-DD HH:MM:SS`. Якщо опустити компонент часу, буде прийнято опівночі. Таким же чином, якщо дата опущена, передбачається поточний день. Наприклад, щоб переглянути повідомлення, зареєстровані з 19:00 до 19:01, ви введете:

```
root@debian:~# journalctl --since "19:00:00" --until "19:01:00"
-- Logs begin at Sat 2019-10-12 18:55:41 CEST, end at Sat 2019-10-12 20:10:50 CEST. --
Oct 12 19:00:14 debian systemd[1]: Started Run anacron jobs.
Oct 12 19:00:14 debian anacron[1057]: Anacron 2.3 started on 2019-10-12
Oct 12 19:00:14 debian anacron[1057]: Normal exit (0 jobs run)
Oct 12 19:00:14 debian systemd[1]: anacron.timer: Adding 2min 47.988096s random time.
```

Так само ви можете використати дещо іншу специфікацію часу: `"integer time-unit ago"`. Таким чином, щоб побачити повідомлення, зареєстровані дві хвилини тому, ви введете `sudo journalctl --since "2 minutes ago"`. Також можна використовувати `+` і `-`, щоб вказати час відносно поточного часу, тому `--since "-2 minutes"` та `--since "2 minutes ago"` є еквівалентними.

Крім числових виразів, ви можете вказати кілька ключових слів:

`yesterday`
: Станом на північ дня перед поточним днем.

`today`
: Станом на північ поточного дня.

`tomorrow`
: Станом на північ наступного дня після поточного дня.

`now`
: Поточний час.

Давайте переглянемо всі повідомлення з опівночі до сьогодні о 21:00:

```
root@debian:~# journalctl --since "today" --until "21:00:00"
-- Logs begin at Sat 2019-10-12 20:45:29 CEST, end at Sat 2019-10-12 21:06:15 CEST. --
Oct 12 20:45:29 debian sudo[1416]:    carol : TTY=pts/0 ; PWD=/home/carol ; USER=root
; COMMAND=/bin/systemctl r
Oct 12 20:45:29 debian sudo[1416]: pam_unix(sudo:session): session opened for user
root by carol(uid=0)
Oct 12 20:45:29 debian systemd[1]: Stopped Flush Journal to Persistent Storage.
(...)
```

NOTE Щоб дізнатися більше про різні синтаксиси специфікацій часу, зверніться до man-сторінки `systemd.time`.

Програма

Щоб переглянути повідомлення журналу, пов'язані з певним виконуваним файлом, використовується такий синтаксис: `journalctl /path/to/executable`:

```
root@debian:~# journalctl /usr/sbin/sshd
-- Logs begin at Sat 2019-10-12 20:45:29 CEST, end at Sat 2019-10-12 21:54:49 CEST. --
Oct 12 21:16:28 debian sshd[1569]: Accepted password for carol from 192.168.1.65 port
34050 ssh2
Oct 12 21:16:28 debian sshd[1569]: pam_unix(sshd:session): session opened for user carol
by (uid=0)
Oct 12 21:16:54 debian sshd[1590]: Accepted password for carol from 192.168.1.65 port
34052 ssh2
Oct 12 21:16:54 debian sshd[1590]: pam_unix(sshd:session): session opened for user carol
```

```
by (uid=0)
```

Модуль

Пам'ятайте, що модулем є будь-який ресурс, який обробляє `systemd`, і за ним ви також можете фільтрувати.

`-u`

Він показує повідомлення про вказаний модуль:

```
root@debian:~# journalctl -u ssh.service
-- Logs begin at Sun 2019-10-13 10:50:59 CEST, end at Sun 2019-10-13 12:22:59 CEST. --
Oct 13 10:51:00 debian systemd[1]: Starting OpenBSD Secure Shell server...
Oct 13 10:51:00 debian sshd[409]: Server listening on 0.0.0.0 port 22.
Oct 13 10:51:00 debian sshd[409]: Server listening on :: port 22.
(...)
```

NOTE Щоб надрукувати всі завантажені та активні модулі, використовуйте `systemctl list-units`; щоб переглянути всі встановлені файли модулів, використовуйте `systemctl list-unit-files`.

Поля

Журнал також можна відфільтрувати за певними *полями* за допомогою будь-якого з наступних синтаксисів:

- `<field-name>=<value>`
- `_<field-name>=<value>_`
- `__<field-name>=<value>`

`PRIORITY=`

Одне з восьми можливих значень пріоритету `syslog` у десятковому форматі:

```
root@debian:~# journalctl PRIORITY=3
-- Logs begin at Sun 2019-10-13 10:50:59 CEST, end at Sun 2019-10-13 14:30:50 CEST.
--
Oct 13 10:51:00 debian avahi-daemon[314]: chroot.c: open() failed: No such file or
directory
```

Зверніть увагу, як ви могли б досягти таких самих вихідних даних, використовуючи команду `sudo journalctl -p err`, яку ми бачили вище.

`SYSLOG_FACILITY=`

Будь-який із можливих кодів об'єкта, відформатований як десятковий рядок. Наприклад, щоб переглянути всі повідомлення на рівні користувача:

```
root@debian:~# journalctl SYSLOG_FACILITY=1
-- Logs begin at Sun 2019-10-13 10:50:59 CEST, end at Sun 2019-10-13 14:42:52 CEST.
--
Oct 13 10:50:59 debian mtp-probe[227]: checking bus 1, device 2:
"/sys/devices/pci0000:00/0000:00:06.0/usb1/1-1"
Oct 13 10:50:59 debian mtp-probe[227]: bus: 1, device: 2 was not an MTP device
Oct 13 10:50:59 debian mtp-probe[238]: checking bus 1, device 2:
"/sys/devices/pci0000:00/0000:00:06.0/usb1/1-1"
Oct 13 10:50:59 debian mtp-probe[238]: bus: 1, device: 2 was not an MTP device
```

`_PID=`

Показати повідомлення, створені певним ідентифікатором процесу. Щоб побачити всі повідомлення, створені `systemd`, вам слід ввести:

```
root@debian:~# journalctl _PID=1
-- Logs begin at Sun 2019-10-13 10:50:59 CEST, end at Sun 2019-10-13 14:50:15 CEST.
--
Oct 13 10:50:59 debian systemd[1]: Mounted Debug File System.
Oct 13 10:50:59 debian systemd[1]: Mounted POSIX Message Queue File System.
Oct 13 10:50:59 debian systemd[1]: Mounted Huge Pages File System.
Oct 13 10:50:59 debian systemd[1]: Started Remount Root and Kernel File Systems.
Oct 13 10:50:59 debian systemd[1]: Starting Flush Journal to Persistent Storage...
(...)
```

`_BOOT_ID`

На основі ідентифікатора завантаження ви можете виділити повідомлення від певного завантаження, наприклад: `sudo journalctl _BOOT_ID=83df3e8653474ea5aed19b41cdb45b78`.

`_TRANSPORT`

Показати повідомлення, отримані від певного транспорту. Можливі значення: `audit` (підсистема аудиту ядра), `driver` (генерується внутрішньо), `syslog` (сокет системного журналу), `journal` (власний протокол журналу), `stdout` (стандартний вихід служб або стандартна помилка) , `kernel` (кільцевий буфер ядра — те саме, що `dmesg`, `journalctl -k` або `journalctl --dmesg`):

```
root@debian:~# journalctl _TRANSPORT=journal
-- Logs begin at Sun 2019-10-13 20:19:58 CEST, end at Sun 2019-10-13 20:46:36 CEST.
--
Oct 13 20:19:58 debian systemd[1]: Started Create list of required static device
nodes for the current kernel.
Oct 13 20:19:58 debian systemd[1]: Starting Create Static Device Nodes in /dev...
Oct 13 20:19:58 debian systemd[1]: Started Create Static Device Nodes in /dev.
Oct 13 20:19:58 debian systemd[1]: Starting udev Kernel Device Manager...
(...)
```

Об'єднання полів

Поля не є взаємовиключними, тому ви можете використовувати декілька в одному запиті. Однак відображатимуться лише повідомлення, що відповідають значенню обох полів одночасно:

```
root@debian:~# journalctl PRIORITY=3 SYSLOG_FACILITY=0
-- No entries --
root@debian:~# journalctl PRIORITY=4 SYSLOG_FACILITY=0
-- Logs begin at Sun 2019-10-13 20:19:58 CEST, end at Sun 2019-10-13 20:21:55 CEST. --
Oct 13 20:19:58 debian kernel: acpi PNP0A03:00: fail to add MMCONFIG information, can't
access extended PCI configuration (...)
```

Якщо ви не використовуєте роздільник `+` для поєднання двох виразів у спосіб логічного *АБО*:

```
root@debian:~# journalctl PRIORITY=3 + SYSLOG_FACILITY=0
-- Logs begin at Sun 2019-10-13 20:19:58 CEST, end at Sun 2019-10-13 20:24:02 CEST. --
Oct 13 20:19:58 debian kernel: Linux version 4.9.0-9-amd64 (debian-kernel@lists.debian.org)
(...9
Oct 13 20:19:58 debian kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.9.0-9-amd64
root=UUID= (...)
(...)
```

З іншого боку, ви можете вказати два значення для одного поля, і всі записи, які відповідають будь-якому значенню, будуть показані:

```
root@debian:~# journalctl PRIORITY=1
-- Logs begin at Sun 2019-10-13 17:16:24 CEST, end at Sun 2019-10-13 17:30:14 CEST. --
-- No entries --
```

```
root@debian:~# journalctl PRIORITY=1 PRIORITY=3
-- Logs begin at Sun 2019-10-13 17:16:24 CEST, end at Sun 2019-10-13 17:32:12 CEST. --
Oct 13 17:16:27 debian connmand[459]: __connman_inet_get_pnp_nameservers: Cannot read /pro
Oct 13 17:16:27 debian connmand[459]: The name net.connman.vpn was not provided by any .se
```

NOTE Поля журналу належать до будь-якої з наступних категорій: “Поля журналу користувача”, “Надійні поля журналу”, “Поля журналу ядра”, “Поля від імені іншої програми” та “Поля адреси”. Для отримання додаткової інформації на цю тему, включаючи повний список полів, перегляньте man-сторінку для `systemd.journal-fields(7)`.

Ручні записи в системному журналі: `systemd-cat`

Подібно до того, як команда `logger` використовується для надсилання повідомлень із командного рядка до системного журналу (як ми бачили в попередньому уроці), команда `systemd-cat` виконує подібні, але більш складні задачі з системним журналом. Це дозволяє нам надсилати стандартний вхід (*stdin*), вихід (*stdout*) і помилку (*stderr*) до журналу.

Якщо команду викликати без параметрів, вона надішле все, що прочитає з *stdin*, до журналу. Після завершення натисніть `Ctrl` + `C`:.

```
carol@debian:~$ systemd-cat
This line goes into the journal.
^C
```

Якщо їй передано результат попередньої команди, це також буде надіслано до журналу:

```
carol@debian:~$ echo "And so does this line." | systemd-cat
```

Якщо за нею слідує команда, результати цієї команди також будуть надіслані до журналу разом із *stderr* (якщо є):

```
carol@debian:~$ systemd-cat echo "And so does this line too."
```

Також існує можливість вказати рівень пріоритету за допомогою опції `-p`:

```
carol@debian:~$ systemd-cat -p emerg echo "This is not a real emergency."
```

Зверніться до man-сторінки `systemd-cat`, щоб дізнатися про інші параметри.

Щоб переглянути останні чотири рядки в журналі:

```
carol@debian:~$ journalctl -n 4
(...)
-- Logs begin at Sun 2019-10-20 13:43:54 CEST. --
Nov 13 23:14:39 debian cat[1997]: This line goes into the journal.
Nov 13 23:19:16 debian cat[2027]: And so does this line.
Nov 13 23:23:21 debian echo[2030]: And so does this line too.
Nov 13 23:26:48 debian echo[2034]: This is not a real emergency.
```

NOTE Записи журналу з рівнем пріоритету *emergency* будуть надруковані жирним червоним шрифтом у більшості систем.

Постійне зберігання журналу

Як згадувалося раніше, у вас є три варіанти щодо розташування журналу:

- Ведення журналу можна взагалі вимкнути (проте все ще можливе переспрямування на інші об'єкти, такі як консоль).
- Зберігайте його в пам'яті, що робить його непостійним, і позбавляйтеся від журналів під час кожного перезавантаження системи. У цьому випадку буде створено та використано каталог `/run/log/journal`.
- Зробіть його постійним, щоб вести журнал на диску. У цьому випадку повідомлення журналу потраплять до каталогу `/var/log/journal`.

Поведінка за замовчуванням така: якщо `/var/log/journal/` не існує, журнали будуть збережені в непостійний спосіб у каталозі `/run/log/journal/` і, отже втрачені з перезавантаженням. Ім'я каталогу — `/etc/machine-id` — це шістнадцятковий 32-символьний рядок нижнього регістру, який закінчується символом переходу до нового рядка:

```
carol@debian:~$ ls /run/log/journal/8821e1fdf176445697223244d1dfbd73/
system.journal
```

Якщо ви спробуєте прочитати його за допомогою less, ви отримаєте попередження, тому замість згаданої команди використовуйте команду `journalctl`:

```
root@debian:~# less /run/log/journal/9a32ba45ce44423a97d6397918de1fa5/system.journal
```

```
"/run/log/journal/9a32ba45ce44423a97d6397918de1fa5/system.journal" may be a binary file.
See it anyway?
root@debian:~# journalctl
-- Logs begin at Sat 2019-10-05 21:26:38 CEST, end at Sat 2019-10-05 21:31:27 CEST. --
(...)
Oct 05 21:26:44 debian systemd-journald[1712]: Runtime journal
(/run/log/journal/9a32ba45ce44423a97d6397918de1fa5) is 4.9M, max 39.5M, 34.6M free.
Oct 05 21:26:44 debian systemd[1]: Started Journal Service.
(...)
```

Якщо `/var/log/journal/` існує, журнали зберігатимуться там постійно. Якщо цей каталог буде видалено, `systemd-journald` не створить його заново, а натомість запише до `/run/log/journal`. Щойно ми знову створимо `/var/log/journal/` і перезапустимо демон, постійне журналювання буде відновлено:

```
root@debian:~# mkdir /var/log/journal/
root@debian:~# systemctl restart systemd-journald
root@debian:~# journalctl
(...)
Oct 05 21:33:49 debian systemd-journald[1712]: Received SIGTERM from PID 1 (systemd).
Oct 05 21:33:49 debian systemd[1]: Stopped Journal Service.
Oct 05 21:33:49 debian systemd[1]: Starting Journal Service...
Oct 05 21:33:49 debian systemd-journald[1768]: Journal started
Oct 05 21:33:49 debian systemd-journald[1768]: System journal
(/var/log/journal/9a32ba45ce44423a97d6397918de1fa5) is 8.0M, max 1.1G, 1.1G free.
Oct 05 21:33:49 debian systemd[1]: Started Journal Service.
Oct 05 21:33:49 debian systemd[1]: Starting Flush Journal to Persistent Storage...
(...)
```

NOTE

За замовчуванням для кожного користувача, який увійшов у систему, будуть окремі файли журналу, розташовані в `/var/log/journal/`, тому разом із файлами `system.journal` ви також знайдете файли типу `user-1000.journal`.

Окрім того, спосіб роботи демона журналу зі збереженням журналів можна змінити після встановлення, налаштувавши його файл конфігурації: `/etc/systemd/journald.conf`. Ключовою опцією є `Storage=` і вона може мати такі значення:

`Storage=volatile`

Дані журналу зберігатимуться виключно в пам'яті у `/run/log/journal/`. Якщо його немає, цей каталог буде створено.

`Storage=persistent`

За замовчуванням дані журналу зберігаються на диску — у папці `/var/log/journal/` — із резервним поверненням до пам'яті (`/run/log/journal/`) під час початкових етапів завантаження та якщо диск недоступний для запису . За потреби буде створено обидва каталоги.

`Storage=auto`

`auto` подібний до `persistent`, але каталог `/var/log/journal` не створюється за потреби. Це значення за умовчанням.

`Storage=none`

Усі дані журналу буде видалено. Однак пересилання до інших цілей, таких як консоль, буфер журналу ядра або сокет системного журналу, все ще можливе.

Наприклад, щоб `systemd-journald` створив `/var/log/journal/` і переключився на постійне сховище, ви повинні відредагувати `/etc/systemd/journald.conf` і встановити `Storage=persistent`, зберегти файл і перезапустіть демон за допомогою `sudo systemctl restart systemd-journald`. Щоб переконатися, що перезапуск пройшов бездоганно, ви завжди можете перевірити статус демона:

```
root@debian:~# systemctl status systemd-journald
 systemd-journald.service - Journal Service
   Loaded: loaded (/lib/systemd/system/systemd-journald.service; static; vendor preset:
enabled)
   Active: active (running) since Wed 2019-10-09 10:03:40 CEST; 2s ago
     Docs: man:systemd-journald.service(8)
           man:journald.conf(5)
 Main PID: 1872 (systemd-journal)
   Status: "Processing requests..."
    Tasks: 1 (limit: 3558)
   Memory: 1.1M
   CGroup: /system.slice/systemd-journald.service
           └─1872 /lib/systemd/systemd-journald

Oct 09 10:03:40 debian10 systemd-journald[1872]: Journal started
Oct 09 10:03:40 debian10 systemd-journald[1872]: System journal
(/var/log/journal/9a32ba45ce44423a97d6397918de1fa5) is 8.0M, max 1.2G, 1.2G free.
```

NOTE

Файли журналу в `/var/log/journal/<machine-id>/` або `/run/log/journal/<machine-id>/` мають суфікс `.journal` (наприклад, `system.journal`) . Однак, якщо буде виявлено, що вони пошкоджені або

> демон зупинено неналежним чином, їх буде перейменовано з додаванням `~` (наприклад, `system.journal~`), і демон почне писати до нового чистого файлу.

Видалення старих даних журналу. Розмір журналу

Журнали зберігаються у *файлах журналу*, імена файлів яких закінчуються на `.journal` або `.journal~` і знаходяться у відповідному каталозі (`/run/log/journal` або `/var/log/journal`, залежно від налаштувань). Щоб перевірити, скільки дискового простору зараз зайнято файлами журналу (як заархівованими, так і активними), використовуйте перемикач `--disk-usage`:

```
root@debian:~# journalctl --disk-usage
Archived and active journals take up 24.0M in the filesystem.
```

Журнали `systemd` за замовчуванням можуть використовувати максимум 10% від розміру файлової системи, де вони зберігаються. Наприклад, у файловій системі 1 ГБ вони не займатимуть більше 100 МБ. Після досягнення цього ліміту старі журнали почнуть зникати, щоб зайнятий простір залишався наближеним до цього значення.

Однак обмеження розміру збережених файлів журналу можна контролювати, налаштувавши низку параметрів конфігурації в `/etc/systemd/journald.conf`. Ці параметри поділяються на дві категорії залежно від типу файлової системи, яка використовується: постійні (`/var/log/journal`) або в пам`яті (`/run/log/journal`). У першому варіанті використовуються параметри, які починаються словом `System` і застосовуватимуться, лише якщо постійне журналювання ввімкнено належним чином і після повного завантаження операційної системи. Назви параметрів для другої категорії починаються зі слова `Runtime` і застосовуватимуться в таких сценаріях:

`SystemMaxUse=`, **`RuntimeMaxUse=`**

Вони контролюють обсяг дискового простору, який може зайняти журнал. За замовчуванням він становить 10% від розміру файлової системи, але його можна змінити (наприклад, `SystemMaxUse=500M`), якщо значення не перевищує максимум 4 ГБ.

`SystemKeepFree=`, **`RuntimeKeepFree=`**

Вони контролюють обсяг дискового простору, який має залишатися вільним для інших користувачів. За замовчуванням він становить 15% від розміру файлової системи, але його можна змінити (наприклад, `SystemKeepFree=500M`), якщо значення не перевищує максимум 4 ГБ.

Щодо пріоритету `*MaxUse` і `*KeepFree`, `systemd-journald` задовольнить обидва, використовуючи менше з двох значень. Так само майте на увазі, що видаляються лише архівні (на відміну від активних) файли журналу.

`SystemMaxFileSize=`, `RuntimeMaxFileSize=`

Вони контролюють максимальний розмір, до якого можуть зрости окремі файли журналу. Типовим значенням є 1/8 від `*MaxUse`. Зменшення розміру виконується синхронно, і значення можна вказувати в байтах або за допомогою `K`, `M`, `G`, `T`, `P`, `E` для кілобайт, мегабайт, гігабайт, терабайт, петабайт і ексібайт відповідно.

`SystemMaxFiles=`, `RuntimeMaxFiles=`

Вони встановлюють максимальну кількість окремих і архівних файлів журналу для зберігання (на активні файли журналу не впливають). За замовчуванням 100.

Окрім видалення на основі розміру та ротації повідомлень журналу, `systemd-journald` також дозволяє використовувати критерії на основі часу, використовуючи такі два параметри: `MaxRetentionSec=` і `MaxFileSec=`. Зверніться до man-сторінки `journald.conf` для отримання додаткової інформації про ці та інші параметри.

NOTE

Кожного разу, коли ви змінюєте типову поведінку `systemd-journald` шляхом розкоментування та редагування параметрів у `/etc/systemd/journald.conf`, ви повинні перезапустити демон, щоб зміни набули чинності.

Очищення журналу

Ви можете вручну очистити архівні файли журналу в будь-який час за допомогою будь-якого з наступних трьох варіантів:

`--vacuum-time=`

This time-based option will eliminate all messages in journal files with a timestamp older than the specified timeframe. Values must be written with any of the following suffixes: `s`, `m`, `h`, `days` (or `d`), `months`, `weeks` (or `w`) and `years` (or `y`). For instance, to get rid of all messages in archived journal files that are older than 1 month:

```
root@debian:~# journalctl --vacuum-time=1months
Deleted archived journal
/var/log/journal/7203088f20394d9c8b252b64a0171e08/system@27dd08376f71405a91794e632ede97ed
-0000000000000001-00059475764d46d6.journal (16.0M).
Deleted archived journal /var/log/journal/7203088f20394d9c8b252b64a0171e08/user-
1000@e7020d80d3af42f0bc31592b39647e9c-000000000000008e-00059479df9677c8.journal (8.0M).
```

`--vacuum-size=`

This size-based option will delete archived journal files until they occupy a value below the specified size. Values must be written with any of the following suffixes: `K`, `M`, `G` or `T`. For instance, to eliminate archived journal files until they are below 100 Mebibytes:

```
root@debian:~# journalctl --vacuum-size=100M
Vacuuming done, freed 0B of archived journals from
/run/log/journal/9a32ba45ce44423a97d6397918de1fa5.
```

`--vacuum-files=`

This option will take care that no more archived journal files than the specified number remain. The value is an integer. For instance, to limit the number of archived journal files to 10:

```
root@debian:~# journalctl --vacuum-files=10
Vacuuming done, freed 0B of archived journals from
/run/log/journal/9a32ba45ce44423a97d6397918de1fa5.
```

Під час очищення видаляються лише архівні файли журналу. Якщо ви хочете позбутися всього (включно з активними файлами журналу), вам потрібно використати сигнал (`SIGUSR2`), який викликає негайну ротацію файлів журналу з опцією `--rotate`. Інші важливі сигнали можна викликати за допомогою таких параметрів:

`--flush` (`SIGUSR1`)

Він використовується для очищення файлів журналу з `/run/` до `/var/`, щоб зробити журнал постійним. Для цього потрібно ввімкнути постійне журналювання та змонтувати `/var/`.

`--sync` (`SIGRTMIN+1`)

Він використовується для запиту запису всіх незаписаних даних журналу на диск.

NOTE

Щоб перевірити внутрішню узгодженість файлу журналу, використовуйте `journalctl` з опцією `--verify`. Коли перевірка буде виконана, ви побачите індикатор перебігу та всі можливі проблеми.

Отримання даних журналу із системи відновлення

Як системний адміністратор ви можете опинитися в ситуації, коли вам знадобиться отримати доступ до файлів журналу на жорсткому диску несправної машини через

систему відновлення (завантажувальний компакт-диск або USB-ключ, що містить живий дистрибутив Linux).

`journalctl` шукає файли журналу в `/var/log/journal/<machine-id>/`. Оскільки ідентифікатори машин у резервній та несправній системах відрізнятимуться, ви повинні використати такий параметр:

`-D </path/to/dir>`, **`--directory=</path/to/dir>`**

За допомогою цього параметра ми вказуємо шлях до каталогу, де `journalctl` шукатиме файли журналу замість типових місць виконання та системних розташувань.

Таким чином, вам необхідно змонтувати `rootfs` несправної системи (`/dev/sda1`) у файловій резервній системі та продовжити читання файлів журналу таким чином:

```
root@debian:~# journalctl -D /media/carol/faulty.system/var/log/journal/
-- Logs begin at Sun 2019-10-20 12:30:45 CEST, end at Sun 2019-10-20 12:32:57 CEST. --
oct 20 12:30:45 suse-server kernel: Linux version 4.12.14-lp151.28.16-default
(geeko@buildhost) (...)
oct 20 12:30:45 suse-server kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-4.12.14-
lp151.28.16-default root=UUID=7570f67f-4a08-448e-aa09-168769cb9289 splash=>
oct 20 12:30:45 suse-server kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating
point registers'
oct 20 12:30:45 suse-server kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
(...)
```

Інші параметри, які можуть бути корисними в цьому випадку:

`-m`, **`--merge`**

Об’єднує записи з усіх доступних журналів у `/var/log/journal`, включаючи віддалені.

`--file`

Покаже записи в певному файлі, наприклад: `journalctl --file /var/log/journal/64319965bda04dfa81d3bc4e7919814a/user-1000.journal`.

`--root`

Шлях до каталогу, що означає, що кореневий каталог передається як аргумент. `journalctl` шукатиме там файли журналу (наприклад, `journalctl --root /faulty.system/`).

Для отримання додаткової інформації перегляньте man-сторінку `journalctl`.

Пересилання даних журналу до традиційного демона `syslog`

Дані журналу можна зробити доступними для традиційного демона `syslog`:

- Пересилання повідомлень до файлу сокета `/run/systemd/journal/syslog` для читання `syslogd`. Цей засіб увімкнено за допомогою параметра `ForwardToSyslog=yes`.
- Наявність демона `syslog`, який поводиться як `journalctl`, тому читає повідомлення журналу безпосередньо з файлів журналу. У цьому випадку відповідним параметром є `Storage`; він повинен мати значення, відмінне від `none`.

NOTE

Так само ви можете пересилати повідомлення журналу в інші місця призначення за допомогою таких параметрів: `ForwardToKMsg` (буфер журналу ядра — `kmsg`), `ForwardToConsole` (системна консоль) або `ForwardToWall` (усі користувачі, які ввійшли в систему через `wall`). Для отримання додаткової інформації зверніться до man-сторінки для `journald.conf`.

Вправи до посібника

1. Припускаючи, що ви `root`, заповніть таблицю відповідною командою `journalctl`:

Призначення	**Команда**
Друкувати записи ядра	
Друкувати повідомлення з другого завантаження, починаючи з початку журналу	
Друкувати повідомлення з другого завантаження, починаючи з кінця журналу	
Друкувати останні повідомлення журналу та стежити за новими	
Відтепер друкувати лише нові повідомлення та постійно оновлювати вивід	
Друкувати повідомлення з попереднього завантаження з пріоритетом `warning` та у зворотному порядку	

2. Поведінка демона журналу щодо зберігання в основному контролюється значенням опції `Storage` у `/etc/systemd/journald.conf`. У наведеній нижче таблиці вкажіть, яка поведінка пов'язана з яким значенням:

Поведінка	`Storage=auto`	`Storage=none`	`Storage=persistent`	`Storage=volatile`
Дані журналу викидаються, але пересилання можливе.				

Поведінка	`Storage=auto`	`Storage=none`	`Storage=persistent`	`Storage=volatile`
Після завантаження системи дані журналу будуть збережені в папці `/var/log/journal`. Буде створено каталог, якщо його ще немає.				
Після завантаження системи дані журналу будуть збережені в папці `/var/log/journal`. Якщо його ще немає, каталог не буде створено.				
Дані журналу зберігатимуться в `/var/run/journal`, але не переживуть перезавантаження.				

3. Як ви дізналися, журнал можна очищати вручну в залежності від часу, розміру та кількості файлів. Виконайте наступні завдання за допомогою `journalctl` і відповідних параметрів:
   - Перевірте, скільки місця на диску займають файли журналу:

- Зменшить кількість місця, зарезервованого для архівних файлів журналу, і встановіть його на 200 МБ:

- Ще раз перевірте простір на диску та поясніть результати:

Дослідницькі вправи

1. Які параметри слід змінити в `/etc/systemd/journald.conf`, щоб повідомлення пересилалися до `/dev/tty5`? Які значення повинні мати параметри?

2. Укажіть правильний фільтр `journalctl`, щоб надрукувати наступне:

Призначення	**Фільтр + значення**
Друкувати повідомлення, що належать певному користувачеві	
Друкувати повідомлення з хоста під назвою `debian`	
Друкувати повідомлення, що належать до певної групи	
Друкувати повідомлення, що належать до `root`	
На основі шляху до виконуваного файлу вивести повідомлення `sudo`	
На основі назви команди вивести повідомлення `sudo`	

3. Під час фільтрації за пріоритетом журнали з вищим пріоритетом, ніж зазначено, також будуть включені до списку; наприклад, `journalctl -p err` виведе повідомлення *error*, *critical*, *alert* та *emergency*. Однак ви можете наказати `journalctl` показувати лише певний діапазон. Яку команду ви використаєте, щоб `journalctl` друкував лише повідомлення з рівнями пріоритету *warning*, *error* і *critical*?

4. Рівні пріоритету також можна вказати у числовому представленні. Перепишіть команду з попередньої вправи, використовуючи числове представлення рівнів пріоритету:

Підсумки

В цьому уроці ми вивчили:

- Переваги використання `systemd` як менеджера системи та послуг.
- Основи модулів і цілей `systemd`.
- Звідки `systemd-journald` отримує дані журналу.
- Параметри, які можна передати `systemctl` для керування `systemd-journald`: `start`, `status`, `restart` і `stop`.
- Де знаходиться файл конфігурації журналу `/etc/systemd/journald.conf` і його основні параметри.
- Як запитувати журнал у загальному варіанті і для конкретних даних за допомогою фільтрів.
- Як здійснювати навігацію та пошук у журналі.
- Як працювати зі зберіганням файлів журналу: у пам'яті чи на диску.
- Як взагалі вимкнути ведення журналу.
- Як перевірити дисковий простір, зайнятий журналом, застосувати обмеження на розмір збережених файлів журналу та очистити архівні файли журналу вручну (*vacumming*).
- Як отримати дані журналу з резервної системи.
- Як пересилати дані журналу до традиційного демона `syslog`.

Команди, які використовуються в цьому уроці:

`systemctl`
: Керує системою та менеджером служб `systemd`.

`journalctl`
: Робить запит до журналу `systemd`.

`ls`
: Виводить вміст каталогу.

`less`
: Переглядає вміст файлу.

`mkdir`

Створює каталоги.

Відповіді до вправ посібника

1. Припускаючи, що ви `root`, заповніть таблицю відповідною командою `journalctl`:

Призначення	**Команда**
Друкувати записи ядра	`journalctl -k` або `journalctl --dmesg`
Друкувати повідомлення з другого завантаження, починаючи з початку журналу	`journalctl -b 2`
Друкувати повідомлення з другого завантаження, починаючи з кінця журналу	`journalctl -b -2 -r` або `journalctl -r -b -2`
Друкувати останні повідомлення журналу та стежити за новими	`journalctl -f`
Відтепер друкувати лише нові повідомлення та постійно оновлювати вихід	`journalctl --since "now" -f`
Друкувати повідомлення з попереднього завантаження з пріоритетом `попередження` та у зворотному порядку	`journalctl -b -1 -p попередження -r`

2. Поведінка демона журналу щодо зберігання в основному контролюється значенням опції `Storage` у `/etc/systemd/journald.conf`. У наведеній нижче таблиці вкажіть, яка поведінка пов'язана з яким значенням:

Поведінка	`Storage=auto`	`Storage=none`	`Storage=persistent`	`Storage=volatile`
Дані журналу викидаються, але пересилання можливе		x		

Поведінка	`Storage=auto`	`Storage=none`	`Storage=persistent`	`Storage=volatile`
Після завантаження системи дані журналу будуть збережені в папці `/var/log/journal`. Буде створено каталог, якщо його ще немає.			x	
Після завантаження системи дані журналу будуть збережені в папці `/var/log/journal`. Якщо його ще немає, каталог не буде створено.	x			
Дані журналу зберігатимуться в `/var/run/journal`, але не переживуть перезавантаження				x

3. Як ви дізналися, журнал можна очищати вручну в залежності від часу, розміру та кількості файлів. Виконайте наступні завдання за допомогою `journalctl` і відповідних параметрів:
   - Перевірте, скільки місця на диску займають файли журналу:

```
journalctl --disk-usage
```

- Зменшить кількість місця, зарезервованого для архівних файлів журналу, і встановіть його на 200 МБ:

```
journalctl --vacuum-size=200M
```

- Ще раз перевірте простір на диску та поясніть результати:

```
journalctl --disk-usage
```

Кореляції немає, оскільки `--disk-usage` показує простір, зайнятий як активними, так і архівними файлами журналу, тоді як `--vacuum-size` стосується лише архівних файлів.

Відповіді до дослідницьких вправ

1. Які параметри слід змінити в `/etc/systemd/journald.conf`, щоб повідомлення пересилалися до `/dev/tty5`? Які значення повинні мати параметри?

```
ForwardToConsole=yes
TTYPath=/dev/tty5
```

2. Укажіть правильний фільтр `journalctl`, щоб надрукувати наступне:

Призначення	**Фільтр + значення**
Друкувати повідомлення, що належать певному користувачеві	`_ID=<id-користувача>`
Друкувати повідомлення з хоста під назвою `debian`	`_HOSTNAME=debian`
Друкувати повідомлення, що належать до певної групи	`_GID=<ідентифікатор групи>`
Друкувати повідомлення, що належать до `root`	`_UID=0`
На основі шляху до виконуваного файлу вивести повідомлення `sudo`	`_EXE=/usr/bin/sudo`
На основі назви команди вивести повідомлення `sudo`	`_COMM=sudo`

3. Під час фільтрації за пріоритетом журнали з вищим пріоритетом, ніж зазначено, також будуть включені до списку; наприклад, `journalctl -p err` виведе повідомлення *error*, *critical*, *alert* та *emergency*. Однак ви можете наказати `journalctl` показувати лише певний діапазон. Яку команду ви використаєте, щоб `journalctl` друкував лише повідомлення з рівнями пріоритету *warning*, *error* і *critical*?

```
journalctl -p warning..crit
```

4. Рівні пріоритету також можна вказати у числовому представленні. Перепишіть команду з попередньої вправи, використовуючи числове представлення рівнів пріоритету:

```
journalctl -p 4..2
```

108.3 Основи Mail Transfer Agent (MTA

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 108.3

Ваговий коефіцієнт

3

Ключові галузі знань

- Створення псевдонімів електронної пошти.
- Налаштування пересилань електронної пошти.
- Знання загальнодоступних програм MTA (postfix, sendmail, exim) (без налаштування).

Частковий список файлів, термінів та утиліт, що використовуються

- `~/.forward`
- команди рівня емуляції sendmail
- `newaliases`
- `mail`
- `mailq`
- `postfix`
- `sendmail`
- `exim`

Linux Professional Institute

108.3 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	108 Основні системні служби
Тема:	108.3 Основи Mail Transfer Agent (MTA).
Урок:	1 з 1

Вступ

В Unix-подібних операційних системах, таких як Linux, кожен користувач має свою власну папку *вхідні*: спеціальне місце у файловій системі, яке недоступне для інших користувачів без root-прав і зберігає особисті повідомлення електронної пошти користувача. Нові вхідні повідомлення додаються до папки «Вхідні» користувача за допомогою *Mail Transfer Agent* (MTA). MTA — це програма, що працює як системна служба, яка збирає повідомлення, надіслані іншими локальними обліковими записами, а також повідомлення, отримані з мережі, надіслані з облікових записів віддалених користувачів.

Цей же MTA також відповідає за надсилання повідомлень у мережу, якщо адреса призначення відноситься до віддаленого облікового запису. Це робиться за допомогою розташування файлової системи як електронної скриньки *вихідних повідомлень* для всіх користувачів системи: щойно користувач розміщує нове повідомлення в папці вихідних повідомлень, MTA ідентифікує цільовий мережевий вузол за доменним іменем, указаним адресою електронної пошти призначення, частина після знака @, і тоді він спробує передати повідомлення до віддаленого MTA за допомогою *Simple Mail Transfer Protocol* (SMTP). SMTP розроблено з урахуванням ненадійних мереж, тому він намагатиметься встановити альтернативні маршрути доставки, якщо основний вузол призначення пошти

недоступний.

Локальний і віддалений MTA

Традиційні облікові записи користувачів на машинах, підключених до мережі, разом складають найпростіший сценарій обміну електронною поштою, де кожен вузол мережі запускає власний демон MTA. Для надсилання та отримання повідомлень електронної пошти не потрібне інше програмне забезпечення, окрім MTA. Однак на практиці більш поширеним є використання віддаленого облікового запису електронної пошти без активної локальної служби MTA (тобто натомість для доступу до віддаленого облікового запису використовується клієнтська програма електронної пошти).

На відміну від локальних облікових записів, обліковий запис віддаленої електронної пошти, також званий *віддаленою поштовою скринькою*, вимагає автентифікації користувача, щоб надати доступ до поштової скриньки користувача та до віддаленого MTA (у цьому випадку назвемо його просто *SMTP-сервер*). У той час як користувач, який взаємодіє з локальною папкою «Вхідні» та MTA, вже ідентифікований системою, віддалена система повинна перевірити особу користувача перед обробкою його повідомлень через IMAP або POP3.

NOTE

Зараз найпоширенішим способом надсилання та отримання електронної пошти є обліковий запис, розміщений на віддаленому сервері, наприклад, централізований сервер електронної пошти компанії, на якому розміщені облікові записи всіх співробітників, або особиста служба електронної пошти, наприклад *Gmail* Google. Замість того, щоб збирати локально доставлені повідомлення, клієнтська програма електронної пошти підключатиметься до віддаленої поштової скриньки та отримуватиме повідомлення звідти. Для отримання повідомлень із віддаленого сервера зазвичай використовуються протоколи POP3 та IMAP, але також можуть використовуватися інші нестандартні власні протоколи.

Коли демон MTA працює в локальній системі, локальні користувачі можуть надсилати електронні листи іншим локальним користувачам або користувачам на віддаленій машині, за умови, що в їхній системі також є служба MTA, яка приймає мережеві підключення. TCP-порт 25 є стандартним портом для зв'язку SMTP, але можуть використовуватися й інші порти залежно від схеми автентифікації, яка використовується, та/або схеми шифрування (якщо така є).

Залишаючи осторонь топології, що передбачають доступ до віддалених поштових скриньок, мережа обміну електронною поштою між звичайними обліковими записами користувачів Linux може бути реалізована, якщо всі вузли мережі мають активний MTA,

який здатний виконувати наступні завдання:

- Вести чергу вихідних повідомлень для надсилання. Для кожного повідомлення в черзі локальний MTA оцінюватиме MTA призначення за адресою одержувача.
- Спілкуватися з віддаленими демонами MTA за допомогою SMTP. Локальний MTA повинен мати можливість використовувати простий протокол передачі пошти (SMTP) через стек TCP/IP для отримання, надсилання та перенаправлення повідомлень від/до інших віддалених демонов MTA.
- Підтримувати окрему папку «Вхідні» для кожного локального облікового запису. MTA зазвичай зберігає повідомлення у форматі *mbox*: один текстовий файл, що містить усі повідомлення електронної пошти послідовно.

Зазвичай адреси електронної пошти вказують доменне ім'я як місцезнаходження, наприклад, `lpi.org` в `info@lpi.org`. У цьому випадку MTA відправника надсилатиме запит до служби DNS щодо відповідного запису MX. Запис DNS MX містить IP-адресу MTA, що обробляє електронну пошту для цього домену. Якщо той самий домен має більше ніж один запис MX, указаний у DNS, MTA має спробувати зв'язатися з ними відповідно до їхніх пріоритетів. Якщо адреса одержувача не вказує ім'я домену або домен не має запису MX, то частина після символу `@` розглядатиметься як хост цільового MTA.

Аспекти безпеки повинні бути враховані, якщо хости MTA будуть видимі для хостів в Інтернеті. Наприклад, невідомий користувач може використати локальний MTA, щоб видати себе за іншого користувача та надіслати потенційно шкідливі електронні листи. MTA, який сліпо передає електронну пошту, відомий як *відкритий ретранслятор*, коли його можна використовувати як посередника для потенційного маскування справжнього відправника повідомлення. Щоб запобігти таким зловживанням, рекомендується приймати з'єднання лише з авторизованих доменів і застосовувати безпечну схему автентифікації.

До того ж, існує багато різних реалізацій MTA для Linux, кожна з яких зосереджена на певних аспектах, таких як сумісність, продуктивність, безпека тощо. Тим не менш, усі MTA дотримуватимуться однакових основних принципів і надаватимуть подібні функції.

MTA для Linux

Традиційним MTA, доступним для систем Linux, є *Sendmail*, дуже гнучкий MTA загального призначення, який використовується багатьма Unix-подібними операційними системами. Іншими поширеними MTA є *Postfix*, *qmail* і *Exim*. Основною причиною вибору альтернативного MTA є полегшення впровадження розширених функцій, оскільки налаштування користувацьких серверів електронної пошти в Sendmail може бути

складним завданням. Крім того, кожен дистрибутив може мати свій бажаний MTA із попередньо визначеними параметрами, які відповідають найбільш поширеним налаштуванням. Усі MTA призначені для заміни Sendmail, тому всі програми, сумісні з Sendmail, повинні працювати незалежно від того, який MTA використовується.

Якщо MTA працює, але не приймає мережеві підключення, він зможе доставляти повідомлення електронної пошти лише на локальній машині. Для MTA `sendmail` файл `/etc/mail/sendmail.mc` слід змінити, щоб приймати нелокальні підключення. Для цього запис

```
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

потрібно змінити на правильну мережеву адресу та перезапустити службу. Деякі дистрибутиви Linux, наприклад Debian, можуть пропонувати інструменти конфігурації, які допомагають запустити сервер електронної пошти з попередньо визначеним набором функцій, які часто використовуються.

TIP

Через проблеми безпеки більшість дистрибутивів Linux не встановлюють MTA за замовчуванням. Щоб перевірити приклади, наведені в цьому уроці, переконайтеся, що на кожній машині запущений MTA і що вони приймають з'єднання через TCP-порт 25. З метою безпеки ці системи не повинні піддаватися впливу вхідних з'єднань із загальнодоступного Інтернету під час тестування.

Коли MTA запущено та приймає з'єднання з мережею, нові повідомлення електронної пошти передаються йому за допомогою команд SMTP, які надсилаються через з'єднання TCP. Команда `nc`, мережева утиліта, яка читає та записує загальні дані в мережі, може використовуватися для надсилання команд SMTP безпосередньо до MTA. Якщо команда `nc` недоступна, її буде встановлено з пакетом *ncat* або *nmap-ncat*, залежно від системи керування пакунками, яка використовується. Написання команд SMTP безпосередньо в MTA допоможе вам краще зрозуміти протокол та інші загальні концепції електронної пошти, але це також може допомогти діагностувати проблеми в процесі доставки електронної пошти.

Якщо, наприклад, користувач `emma` на хості `lab1.campus` хоче надіслати повідомлення користувачеві `dave` на хості `lab2.campus`, тоді вона може використати команду `nc` для прямого підключення до MTA `lab2.campus`, припускаючи, що він прослуховує TCP-порт 25:

```
$ nc lab2.campus 25
220 lab2.campus ESMTP Sendmail 8.15.2/8.15.2; Sat, 16 Nov 2019 00:16:07 GMT
```

```
HELO lab1.campus
250 lab2.campus Hello lab1.campus [10.0.3.134], pleased to meet you
MAIL FROM: emma@lab1.campus
250 2.1.0 emma@lab1.campus... Sender ok
RCPT TO: dave@lab2.campus
250 2.1.5 dave@lab2.campus... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Subject: Recipient MTA Test

Hi Dave, this is a test for your MTA.
.
250 2.0.0 xAG0G7Y0000595 Message accepted for delivery
QUIT
221 2.0.0 lab2.campus closing connection
```

Після встановлення з'єднання віддалений MTA ідентифікує себе та готовий приймати команди SMTP. Перша команда SMTP у прикладі, `HELO lab1.campus`, вказує `lab1.campus` як ініціатор обміну. Наступні дві команди, `MAIL FROM: emma@lab1.campus` і `RCPT TO: dave@lab2.campus`, вказують відправника та одержувача. Правильне повідомлення електронної пошти починається після команди `DATA` і закінчується крапкою в окремому рядку. Щоб додати поле `subject` до електронного листа, воно має бути в першому рядку після команди `DATA`, як показано в прикладі. Коли subject-поле використовується, має бути порожній рядок, який відокремлює його від вмісту електронної пошти. Команда `QUIT` завершує з'єднання з MTA на хості `lab2.campus`.

На хості `lab2.campus` користувач `dave` отримає повідомлення, подібне до `You have new mail in /var/spool/mail/dave`, як тільки він увійде в сеанс оболонки. Цей файл міститиме необроблене повідомлення електронної пошти, надіслане `emma`, а також заголовки, додані MTA:

```
$ cat /var/spool/mail/dave
From emma@lab1.campus  Sat Nov 16 00:19:13 2019
Return-Path: <emma@lab1.campus>
Received: from lab1.campus (lab1.campus [10.0.3.134])
    by lab2.campus (8.15.2/8.15.2) with SMTP id xAG0G7Y0000595
    for dave@lab2.campus; Sat, 16 Nov 2019 00:17:06 GMT
Date: Sat, 16 Nov 2019 00:16:07 GMT
From: emma@lab1.campus
Message-Id: <201911160017.xAG0G7Y0000595@lab2.campus>
Subject: Recipient MTA Test
```

```
Hi Dave, this is a test for your MTA.
```

Заголовок `Received:` показує, що повідомлення від `lab1.campus` було отримано безпосередньо `lab2.campus`. За замовчуванням MTA прийматимуть повідомлення лише для локальних одержувачів. Якщо користувач `emma` намагатиметься надіслати електронний лист користувачеві `henry` на хості `lab3.campus`, але використовуючи MTA `lab2.campus` замість належного MTA `lab3.campus`, може виникнути така помилка:

```
$ nc lab2.campus 25
220 lab2.campus ESMTP Sendmail 8.15.2/8.15.2; Sat, 16 Nov 2019 00:31:44 GMT
HELO lab1.campus
250 lab2.campus Hello lab1.campus [10.0.3.134], pleased to meet you
MAIL FROM: emma@lab1.campus
250 2.1.0 emma@lab1.campus... Sender ok
RCPT TO: henry@lab3.campus
550 5.7.1 henry@lab3.campus... Relaying denied
```

Номери відповіді SMTP, що починаються з 5, як повідомлення `Relaying denied`, вказують на помилку. Існують законні ситуації, коли ретрансляція бажана, наприклад, коли хости, які надсилають і отримують електронні листи, не підключені весь час: проміжний MTA можна налаштувати для прийому електронних листів, призначених іншим хостам, діючи як *relay* SMTP-сервер, який може пересилати повідомлення між MTA .

Можливість маршрутизації трафіку електронної пошти через проміжні SMTP-сервери перешкоджає спробам підключитися безпосередньо до хоста, указаного електронною адресою одержувача, як показано в попередніх прикладах. Крім того, адреси електронної пошти часто мають доменне ім'я як місце розташування (після `@`), тому фактичне ім'я відповідного хосту MTA має бути отримано через DNS. Тому, якщо використовуються віддалені поштові скриньки, рекомендується делегувати завдання визначення відповідного вузла призначення локальному MTA або віддаленому серверу SMTP.

Sendmail надає команду `sendmail` для виконання багатьох операцій, пов'язаних з електронною поштою, включно з допомогою у створенні нових повідомлень. Він також вимагає від користувача вводити заголовки електронної пошти вручну, але більш зручним способом, ніж безпосередньо використовувати команди SMTP. Отже, більш адекватним способом для користувача `emma@lab1.campus` надіслати повідомлення електронної пошти на `dave@lab2.campus` буде:

```
$ sendmail dave@lab2.campus
From: emma@lab1.campus
```

```
To: dave@lab2.campus
Subject: Sender MTA Test

Hi Dave, this is a test for my MTA.
.
```

Знову ж таки, крапка в рядку сама по собі завершує повідомлення. Повідомлення має бути негайно надіслано одержувачу, хіба що локальний MTA не зможе зв'язатися з віддаленим MTA. Команда `mailq`, якщо її виконує root, покаже всі недоставлені повідомлення. Якщо, наприклад, MTA на `lab2.campus` не відповів, тоді команда `mailq` виведе недоставлене повідомлення та причину збою:

```
# mailq
        /var/spool/mqueue (1 request)
-----Q-ID----- --Size-- -----Q-Time----- ------------Sender/Recipient-----------
xAIK3D9S000453       36 Mon Nov 18 20:03 <emma@lab1.campus>
                 (Deferred: Connection refused by lab2.campus.)
                     <dave@lab2.campus>
        Total requests: 1
```

Типовим розташуванням для черги вихідних повідомлень є `/var/spool/mqueue/`, але різні MTA можуть використовувати різні розташування в каталозі `/var/spool/`. Postfix, наприклад, створить дерево каталогів у `/var/spool/postfix/` для керування чергою. Команда `mailq` еквівалентна `sendmail -bp`, і вони повинні бути присутні незалежно від MTA, встановленого в системі. Щоб забезпечити зворотну сумісність, більшість MTA надають ці традиційні команди адміністрування пошти.

Якщо основний хост призначення електронної пошти (коли він надається із запису MX DNS для домену) недоступний, MTA спробує зв'язатися з записами з нижчим пріоритетом (якщо такі вказані). У випадку таких налаштувань, MTA може періодично перевіряти доступність віддалених хостів і виконувати нову спробу доставки. Якщо використовується MTA, сумісний із Sendmail, нова спроба буде виконана негайно за допомогою команди `sendmail -q`.

Sendmail зберігатиме вхідні повідомлення у файлі, названому на ім'я відповідного власника папки «Вхідні», наприклад `/var/spool/mail/dave`. Інші MTA, наприклад Postfix, можуть зберігати вхідні повідомлення електронної пошти в таких місцях, як `/var/mail/dave`, але вміст файлу той самий. У наведеному прикладі команда `sendmail` була використана на хості відправника для створення повідомлення, тому заголовки необроблених повідомлень показують, що електронний лист пройшов додаткові кроки, перш ніж досягнути кінцевого адресата:

```
$ cat /var/spool/mail/dave
From emma@lab1.campus  Mon Nov 18 20:07:39 2019
Return-Path: <emma@lab1.campus>
Received: from lab1.campus (lab1.campus [10.0.3.134])
    by lab2.campus (8.15.2/8.15.2) with ESMTPS id xAIK7clC000432
    (version=TLSv1.3 cipher=TLS_AES_256_GCM_SHA384 bits=256 verify=NOT)
    for <dave@lab2.campus>; Mon, 18 Nov 2019 20:07:38 GMT
Received: from lab1.campus (localhost [127.0.0.1])
    by lab1.campus (8.15.2/8.15.2) with ESMTPS id xAIK3D9S000453
    (version=TLSv1.3 cipher=TLS_AES_256_GCM_SHA384 bits=256 verify=NOT)
    for <dave@lab2.campus>; Mon, 18 Nov 2019 20:03:13 GMT
Received: (from emma@localhost)
    by lab1.campus (8.15.2/8.15.2/Submit) id xAIK0doL000449
    for dave@lab2.campus; Mon, 18 Nov 2019 20:00:39 GMT
Date: Mon, 18 Nov 2019 20:00:39 GMT
Message-Id: <201911182000.xAIK0doL000449@lab1.campus>
From: emma@lab1.campus
To: dave@lab2.campus
Subject: Sender MTA Test

Hi Dave, this is a test for my MTA.
```

Знизу вгору рядки, що починаються з `Received:`, показують маршрут повідомлення. Повідомлення було надіслано користувачем `emma` за допомогою команди `sendmail dave@lab2.campus`, запущеної на `lab1.campus`, як зазначено в першому заголовку `Received:`. Потім, все ще на `lab1.campus`, MTA використовує ESMTPS — надмножину SMTP, яка додає розширення шифрування — для надсилання повідомлення до MTA на `lab2.campus`, як зазначено в останньому (верхньому) заголовку `Received:`.

MTA завершує свою роботу після того, як повідомлення буде збережено в папці «Вхідні» користувача. Звичайним є виконання певного типу фільтрації електронної пошти, наприклад блокування спаму або застосування визначених користувачем правил фільтрації. Ці завдання виконуються сторонніми додатками, які працюють разом з MTA. MTA може, наприклад, викликати утиліту *SpamAssassin*, щоб позначати підозрілі повідомлення за допомогою функцій аналізу тексту.

Хоча це можливо, незручно читати файл поштової скриньки безпосередньо. Натомість рекомендовано використовувати програму-клієнт електронної пошти (наприклад, Thunderbird, Evolution або KMail), яка аналізуватиме файл і відповідним чином керуватиме повідомленнями. Такі програми також пропонують додаткові функції, наприклад, ярлики для типових дій, підкаталоги вхідних повідомлень тощо.

Команда `mail` і поштові агенти користувача (MUA)

Можна написати повідомлення електронної пошти безпосередньо в необробленому форматі, але набагато практичніше використовувати клієнтську програму, також відому як MUA (*Mail User Agent*), щоб пришвидшити процес і уникнути помилок. MUA піклується про роботу «під капотом», тобто клієнт електронної пошти представляє та впорядковує отримані повідомлення та забезпечує правильний зв'язок із MTA після того, як користувач створює електронний лист.

Існує багато різних типів поштових агентів користувача. Настільні програми, такі як *Mozilla Thunderbird* і *Evolution* від Gnome, підтримують як локальні, так і віддалені облікові записи електронної пошти. Навіть інтерфейси *Webmail* можна розглядати як тип MUA, оскільки вони є посередниками у взаємодії між користувачем і базовим MTA. Однак клієнти електронної пошти не обмежуються графічними інтерфейсами: клієнти електронної пошти консолі широко використовуються для доступу до поштових скриньок, не інтегрованих із графічним інтерфейсом, і для автоматизації завдань, пов'язаних з електронною поштою, у сценаріях оболонки.

Спочатку команда Unix `mail` була призначена лише для обміну повідомленнями між локальними користувачами системи (перша команда `mail` датується першою редакцією Unix, випущеною в 1971 році). Коли мережевий обмін електронною поштою став більш помітним, для роботи з новою системою доставки були створені інші програми, які поступово замінили стару програму `mail`.

Зараз найбільш часто використовувану команду `mail` надає пакет *mailx*, який сумісний з усіма сучасними функціями електронної пошти. У більшості дистрибутивів Linux команда `mail` є лише символьним посиланням на команду `mailx`. Інші реалізації, такі як пакунок *GNU Mailutils*, в основному забезпечують ті самі функції, що й `mailx`. Однак між ними є невеликі відмінності, особливо щодо параметрів командного рядка.

Незалежно від реалізації, усі сучасні варіації команди `mail` працюють у двох режимах: *звичайний режим* і *режим надсилання*. Якщо адреса електронної пошти вказана як аргумент команди `mail`, вона перейде в режим надсилання, інакше вона перейде в звичайний режим (читання). У звичайному режимі отримані повідомлення перераховуються з числовим індексом для кожного, тому користувач може звертатися до них окремо під час введення команд в інтерактивному рядку. Команда `print 1` може бути використана, наприклад, для відображення вмісту повідомлення номер 1. Інтерактивні команди можна скорочувати, тому такі команди, як `print`, `delete` або `reply`, можна замінити на `p`, `d` або `r` відповідно. Команда `mail` завжди розглядатиме останнє отримане або останнє переглянуте повідомлення, якщо номер індексу повідомлення пропущено. Команда `quit` або `q` призведе до виходу з програми.

Режим надсилання особливо корисний для надсилання автоматизованих повідомлень електронної пошти. Його можна використовувати, наприклад, для надсилання електронного листа системному адміністратору, якщо сценарій запланованого технічного обслуговування не виконує своє завдання. У режимі надсилання `mail` використовуватиме вміст зі *стандартного введення* як тіло повідомлення:

```
$ mail -s "Maintenance fail" henry@lab3.campus <<<"The maintenance script failed at `date`"
```

У цьому прикладі було додано опцію `-s`, щоб додати поле теми до повідомлення. Тіло повідомлення надавалося перенаправленням *Hereline* на стандартний вхід, але вміст файлу або вихід команди також можна було передати в *stdin* програми. Якщо перенаправлення до стандартного вводу не забезпечує жодного вмісту, то програма чекатиме, поки користувач введе тіло повідомлення. У цьому випадку натискання клавіші `Ctrl` + `D` завершить повідомлення. Команда `mail` негайно завершить роботу після додавання повідомлення до черги вихідних.

Налаштування доставки

За замовчуванням облікові записи електронної пошти в системі Linux пов'язані зі стандартними системними обліковими записами. Наприклад, якщо користувач Carol має ім'я для входу `carol` на хості `lab2.campus`, то її електронна адреса буде `carol@lab2.campus`. Цей зв'язок «один-до-одного» між системними обліковими записами та поштовими скриньками можна розширити за допомогою стандартних методів, наданих у більшості дистрибутивів Linux, зокрема механізму маршрутизації електронної пошти, який забезпечується файлом `/etc/aliases`.

Псевдонім електронної пошти – це "віртуальний" одержувач електронної пошти, чиї отримані повідомлення перенаправляються до наявних локальних поштових скриньок або до інших типів місць зберігання чи обробки повідомлень. Псевдоніми корисні, наприклад, для розміщення повідомлень, надісланих на `postmaster@lab2.campus`, у поштову скриньку Carol, яка є звичайною локальною поштовою скринькою в системі `lab2.campus`. Для цього слід додати рядок `postmaster: carol` до файлу `/etc/aliases` у `lab2.campus`. Після зміни файлу `/etc/aliases` слід виконати команду `newaliases`, щоб оновити базу даних псевдонімів MTA і зробити зміни дійсними. Команди `sendmail -bi` або `sendmail -I` також можна використовувати для оновлення бази даних псевдонімів.

Псевдоніми визначаються по одному на рядок у форматі `<alias>: <destination>`. Окрім звичайних локальних поштових скриньок, позначених відповідним іменем користувача, доступні інші типи адресатів:

- Повний шлях (починається з `/`) до файлу. Повідомлення, надіслані на відповідний псевдонім, будуть додані до файлу.
- Команда для обробки повідомлення. `<destination>` має починатися з вертикальної лінії, а якщо команда містить спеціальні символи (наприклад, пробіли), її потрібно взяти в подвійні лапки. Наприклад, псевдонім `subscribe: |subscribe.sh` у `lab2.campus` пересилатиме всі повідомлення, надіслані на `subscribe@lab2.campus`, на стандартний вхід команди `subscribe.sh`. Якщо sendmail працює в *обмеженому режимі оболонки*, дозволені команди або посилання на них мають бути в `/etc/smrsh/`.
- Включений файл. Один псевдонім може мати кілька адресатів (розділених комами), тому може бути практичніше зберігати їх у зовнішньому файлі. Ключове слово `:include:` має вказувати шлях до файлу, як у `:include:/var/local/destinations`
- Зовнішня адреса. Псевдоніми також можуть пересилати повідомлення на зовнішні адреси електронної пошти.
- Інший псевдонім.

Непривілейований локальний користувач може визначати псевдоніми для своєї електронної пошти, редагуючи файл `.forward` у своєму домашньому каталозі. Оскільки псевдоніми можуть впливати лише на їх власну поштову скриньку, необхідна лише частина `<destination>`. Щоб пересилати всі вхідні електронні листи на зовнішню адресу, наприклад, користувач `dave` у `lab2.campus` може створити такий файл `~/.forward`:

```
$ cat ~/.forward
emma@lab1.campus
```

Він пересилатиме всі повідомлення електронної пошти, надіслані на `dave@lab2.campus`, на `emma@lab1.campus`. Як і у випадку з файлом `/etc/aliases`, до `.forward` можна додати інші правила переспрямування, по одному на рядок. Тим не менш, файл `.forward` має бути доступний для запису лише його власнику, і немає необхідності виконувати команду `newaliases` після його зміни. Файли, які починаються з крапки, не відображаються у переліку звичайних файлів, через що користувач може не знати про активні псевдоніми. Тому під час діагностики проблем із доставкою електронної пошти важливо перевірити, чи файл існує.

Вправи до посібника

1. Без додаткових опцій чи аргументів команда `mail henry@lab3.campus` переходить у режим введення, щоб користувач міг набрати повідомлення для `henry@lab3.campus`. Після завершення повідомлення, натискання яких клавіш закриє режим введення та відправить електронний лист?

2. Яку команду може виконати користувач root, щоб отримати список недоставлених повідомлень, які створені в локальній системі?

3. Як непривілейований користувач може використовувати стандартний метод MTA для автоматичного пересилання всієї своєї вхідної пошти на адресу `dave@lab2.campus`?

Дослідницькі вправи

1. Використовуючи команду `mail`, надану `mailx`, яка команда надішле повідомлення на `emma@lab1.campus` з файлом `logs.tar.gz` як вкладення та результатом команди `uname -a` як тіло електронної пошти?

2. Адміністратор служби електронної пошти хоче відстежувати передачу електронної пошти через мережу, але він не хоче захаращувати свою поштову скриньку тестовими повідомленнями. Як цей адміністратор міг налаштувати загальносистемний псевдонім електронної пошти, щоб перенаправляти всі електронні листи, надіслані користувачеві `test`, у файл `/dev/null`?

3. Яку команду, окрім `newaliases`, можна використати для оновлення бази даних псевдонімів після додавання нового псевдоніма до `/etc/aliases`?

Підсумки

Цей урок розповідав про роль і використання агентів передачі пошти в системах Linux. MTA забезпечує стандартний метод зв'язку між обліковими записами користувачів і може поєднуватися з іншим програмним забезпеченням для надання додаткових функцій. На уроці розглядалися такі теми:

- Поняття про технології, пов'язані з електронною поштою, поштові скриньки та протоколи.
- Як Linux MTA обмінюються повідомленнями через мережу.
- Консольні клієнти електронної пошти та MUA (агенти користувача пошти).
- Псевдоніми та переадресація локальної електронної пошти.

Розглянуті технології, команди та процедури:

- SMTP і пов'язані протоколи.
- MTA доступні для Linux: Sendmail, Postfix, qmail, Exim.
- Команди MTA і MUA: `sendmail` і `mail`.
- Адміністративні файли та команди: `mailq`, `/etc/aliases`, `nevaliases`, `~/.forward`.

Відповіді до вправ посібника

1. Без додаткових опцій чи аргументів команда `mail henry@lab3.campus` переходить у режим введення, щоб користувач міг набрати повідомлення для `henry@lab3.campus`. Після завершення повідомлення, натискання яких клавіш закриє режим введення та відправить електронний лист?

   Натискання `Ctrl` + `D` призведе до закриття програми та відправлення електронного листа.

2. Яку команду може виконати користувач root, щоб отримати список недоставлених повідомлень, які створені в локальній системі?

   Команду `mailq` або `sendmail -bp`.

3. Як непривілейований користувач може використовувати стандартний метод MTA для автоматичного пересилання всієї своєї вхідної пошти на адресу `dave@lab2.campus`?

   Користувач повинен додати `dave@lab2.campus` до `~/.forward`.

Відповіді до дослідницьких вправ

1. Використовуючи команду `mail`, надану `mailx`, яка команда надішле повідомлення на `emma@lab1.campus` з файлом `logs.tar.gz` як вкладення та результатом команди `uname -a` як тіло електронної пошти?

   `uname -a | mail -a logs.tar.gz emma@lab1.campus`

2. Адміністратор служби електронної пошти хоче відстежувати передачу електронної пошти через мережу, але він не хоче захаращувати свою поштову скриньку тестовими повідомленнями. Як цей адміністратор міг налаштувати загальносистемний псевдонім електронної пошти, щоб перенаправляти всі електронні листи, надіслані користувачеві `test`, у файл `/dev/null`?

   Рядок `test: /dev/null` у `/etc/aliases` перенаправлятиме всі повідомлення, надіслані до локальної поштової скриньки `test`, до файлу `/dev/null`.

3. Яку команду, окрім `newaliases`, можна використати для оновлення бази даних псевдонімів після додавання нового псевдоніма до `/etc/aliases`?

   Команду `sendmail -bi` або `sendmail -I`.

108.4 Налаштування принтерів та друк

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 108.4

Ваговий коефіцієнт

2

Ключові галузі знань

- Базова конфігурація CUPS (для локальних і віддалених принтерів).
- Керування чергами друку користувачів.
- Усунення загальних проблем друку.
- Додавання та видалення завдань із налаштованих черг принтера.

Частковий список файлів, термінів та утиліт, що використовуються

- Конфігураційні файли, інструменти та утиліти CUPS
- `/etc/cups/`
- застарілий інтерфейс lpd (`lpr`, `lprm`, `lpq`)

108.4 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	108 Основні системні служби
Тема:	108.4 Налаштування принтерів та друк
Урок:	1 з 1

Вступ

Заяви про “безпаперове суспільство”, викликані появою комп’ютерів, до теперішнього часу виявилися неправдивими. Багато організацій досі покладаються на друковані або “паперові копії” існуючої інформації. Маючи це на увазі, ми бачимо, наскільки важливо для користувача комп’ютера знати, як друкувати з системи, в той час як адміністратор повинен знати, як підтримувати здатність комп’ютера працювати з принтерами.

У Linux, а також у багатьох інших операційних системах програмний стек *Common Unix Printing System* (CUPS) дозволяє друкувати та керувати принтером із комп’ютера. Наведемо сильно спрощену схему того, як файл друкується в Linux за допомогою CUPS:

1. Користувач надсилає файл для друку.
2. Потім демон CUPS, `cupsd` *додає до пулу* завдання друку. Цьому завданню друку CUPS присвоює номер завдання, а також інформацію про те, яка черга друку містить це завдання, а також назву документа для друку.
3. CUPS використовує *фільтри*, встановлені в системі, для створення відформатованого файлу, який може використовувати принтер.

4. Потім CUPS надсилає переформатований файл на принтер для друку.

Ми розглянемо ці кроки більш детально, а також розглянемо, як встановити принтер і керувати ним у Linux.

Служба CUPS

Більшість настільних дистрибутивів Linux матимуть уже встановлені пакунки CUPS. При встановленні Linux з мінімальними параметрами пакунки CUPS можуть бути не встановлені залежно від дистрибутива. Базове встановлення CUPS можна виконати в системі Debian за допомогою наступних дій:

```
$ sudo apt install cups
```

У системах Fedora процес встановлення так само простий. Вам потрібно буде запустити службу CUPS вручну після встановлення у Fedora та інших дистрибутивах на основі Red Hat:

```
$ sudo dnf install cups
...
$ sudo systemctl start cups.service
```

Після завершення встановлення ви можете переконатися, що служба CUPS працює за допомогою команди `systemctl`:

```
$ systemctl status cups.service
● cups.service - CUPS Scheduler
   Loaded: loaded (/lib/systemd/system/cups.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-06-25 14:35:47 EDT; 41min ago
     Docs: man:cupsd(8)
 Main PID: 3136 (cupsd)
    Tasks: 2 (limit: 1119)
   Memory: 3.2M
   CGroup: /system.slice/cups.service
           ├─3136 /usr/sbin/cupsd -l
           └─3175 /usr/lib/cups/notifier/dbus dbus://
```

Як і багато інших демонов Linux, CUPS покладається на набір конфігураційних файлів для своїх операцій. Нижче наведено основні з них, які цікавлять системного адміністратора:

`/etc/cups/cupsd.conf`

Цей файл містить налаштування конфігурації для самої служби CUPS. Якщо ви взагалі знайомі з файлом конфігурації веб-сервера Apache, то файл конфігурації CUPS здається вам дуже схожим, оскільки він використовує дуже подібний синтаксис. Файл `cupsd.conf` містить параметри для таких речей, як керування доступом до різних черг друку, що використовуються в системі, незалежно від того, увімкнено чи ні веб-інтерфейс CUPS, а також рівень журналювання, який використовуватиме демон.

`/etc/printcap`

Це застарілий файл, який використовувався протоколом LPD (*Line Printer Daemon*) до появи CUPS. CUPS все одно створить цей файл у системах для зворотної сумісності, і часто це символьне посилання на `/run/cups/printcap`. Кожен рядок цього файлу містить принтер, до якого система має доступ.

`/etc/cups/printers.conf`

Цей файл містить кожен принтер, налаштований для використання системою CUPS. Кожен принтер і пов'язана з ним черга друку в цьому файлі зазначені між тегами `<Printer></Printer>`. Цей файл містить окремі списки принтерів, які можна знайти в `/etc/printcap`.

WARNING Не можна змінювати файл `/etc/cups/printers.conf` у командному рядку під час роботи служби CUPS.

`/etc/cups/ppd/`

Це не файл конфігурації, а каталог, який містить файли *PostScript Printer Description* (PPD) для принтерів, які їх використовують. Операційні можливості кожного принтера зберігатимуться у файлі PPD (закінчується розширенням `.ppd`). Це звичайні текстові файли в певному форматі.

Служба CUPS також використовує ведення журналу майже так само, як служба Apache 2. Журнали зберігаються в `/var/log/cups/` і містять `access_log`, `page_log` і `error_log`. `access_log` зберігає записи доступу до веб-інтерфейсу CUPS, а також дії, виконані в ньому, наприклад керування принтером. `page_log` зберігає облік завдань друку, які були надіслані в черги друку, якими керує CUPS. `error_log` міститиме повідомлення про невдалі завдання друку та інші помилки, записані веб-інтерфейсом.

Далі ми розглянемо інструменти та утиліти, які використовуються для керування службою CUPS.

Використання веб-інтерфейсу

Як було зазначено раніше, конфігураційний файл `/etc/cups/cupsd.conf` визначає, чи ввімкнено веб-інтерфейс для системи CUPS. Параметр конфігурації виглядає так:

```
# Web interface setting...
WebInterface Yes
```

Якщо веб-інтерфейс увімкнено, CUPS можна керувати з браузера за типовою URL-адресою `http://localhost:631`. За замовчуванням користувач системи може переглядати принтери та черги друку, але будь-яка форма модифікації конфігурації потребує користувача з правами root для автентифікації у веб-службі. Параметр конфігурації у файлі `/etc/cups/cupsd.conf` для обмеження доступу до адміністративних можливостей матиме такий вигляд:

```
# All administration operations require an administrator to authenticate...
<Limit CUPS-Add-Modify-Printer CUPS-Delete-Printer CUPS-Add-Modify-Class CUPS-Delete-Class
CUPS-Set-Default>
  AuthType Default
  Require user @SYSTEM
  Order deny,allow
</Limit>
```

Наведемо пояснення цих елементів:

`AuthType Default`
: використовує базову підказку автентифікації, коли дія потребує root-доступу.

`Require user @SYSTEM`
: вказує, що для операції буде потрібен користувач з правами адміністратора. Можна змінити на `@groupname`, де члени `groupname` можуть керувати службою CUPS, або окремим користувачам можна надати список, як у `Require user carol, tim`.

`Order deny,allow`
: працює так само, як параметр конфігурації Apache 2, де дію заборонено за умовчанням, якщо користувач (або член групи) не автентифікований.

Веб-інтерфейс для CUPS можна вимкнути, спершу зупинивши службу CUPS, змінивши опцію `WebInterface` з `Yes` на `No`, а потім перезапустивши службу CUPS.

Веб-інтерфейс CUPS побудовано як базовий веб-сайт із вкладками навігації для різних розділів системи CUPS. Веб-інтерфейс містить наступні вкладки:

Home

На домашній сторінці буде показано поточну встановлену версію CUPS. Ця сторінка поділяється на інші розділи, такі як:

CUPS for Users

Містить опис CUPS, параметри командного рядка для роботи з принтерами та чергами друку, а також посилання на форум користувачів CUPS.

CUPS for Administrators

Надає посилання в інтерфейсі для встановлення та керування принтерами та посилання на інформацію про роботу з принтерами в мережі.

CUPS for Developers

Надає посилання на розробку для самого CUPS, а також на створення файлів PPD для принтерів.

Administration

Сторінка адміністрування також розбита на розділи:

Printers

Тут адміністратор може додавати нові принтери до системи, знаходити принтери, підключені до системи, і керувати принтерами, які вже встановлені.

Classes

Класи — це механізм, за допомогою якого принтери можна додавати до груп із певними політиками. Наприклад, клас може містити групу принтерів, які належать до певного поверху будівлі, на яких можуть друкувати лише користувачі певного відділу. Інший клас може мати обмеження щодо кількості сторінок, які користувач може надрукувати. Класи не створюються за замовчуванням під час встановлення CUPS і мають бути визначені адміністратором. Це розділ у веб-інтерфейсі CUPS, де можна створювати нові класи та керувати ними.

Jobs

Тут адміністратор може переглянути всі завдання друку, які зараз знаходяться в черзі для всіх принтерів, якими керує CUPS.

Server

Тут адміністратор може вносити зміни до файлу `/etc/cups/cupsd.conf`. Крім того, додаткові параметри конфігурації доступні за допомогою прапорців, таких як дозвіл принтерам, підключеним до CUPS, спільне використання в мережі, розширена автентифікація та дозвіл віддаленого адміністрування принтера.

Classes

Якщо в системі налаштовано класи принтерів, вони будуть виведені на цій сторінці. Кожен клас принтерів матиме опції для керування всіма принтерами в класі одночасно, а також для перегляду всіх завдань, які стоять у черзі для принтерів у цьому класі.

Help

Ця вкладка містить посилання на всю доступну документацію для CUPS, встановленого в системі.

Jobs

Ця вкладка дозволяє шукати окремі завдання друку, а також виводити список усіх поточних завдань друку, якими керує сервер.

Printers

Ця вкладка містить список усіх принтерів, якими наразі керує система, а також короткий огляд стану кожного принтера. Кожен із указаних принтерів можна натиснути, і адміністратор перейде на сторінку, де можна далі керувати окремим принтером. Інформація про принтери на цій вкладці надходить із файлу `/etc/cups/printers.conf`.

Встановлення принтера

Додавання принтера до системи є простим процесом у веб-інтерфейсі CUPS:

1. Натисніть на вкладку **Administration**, а потім кнопку **Add Printer**.
2. На наступній сторінці будуть представлені різні параметри залежно від способу підключення принтера до системи. Якщо це локальний принтер, виберіть найбільш релевантний параметр, наприклад, до якого порту підключено принтер або яке програмне забезпечення стороннього виробника для принтера може бути встановлено. CUPS також спробує виявити принтери, підключені до мережі, і відобразити їх тут. Ви також можете вибрати варіант прямого підключення до мережевого принтера залежно від того, які протоколи мережевого друку підтримує принтер. Виберіть відповідний варіант і натисніть кнопку **Continue**.

3. На наступній сторінці ви зможете вказати ім'я, опис і розташування (наприклад, "back office" або "front desk" тощо) для принтера. Якщо ви бажаєте надати спільний доступ до цього принтера через мережу, ви також можете встановити прапорець для цього параметра. Після введення налаштувань натисніть кнопку **Continue**.
4. На наступній сторінці можна вибрати марку та модель принтера. Це дозволяє CUPS шукати у своїй локально встановленій базі даних найбільш підходящі драйвери та файли PPD для використання з принтером. Якщо у вас є файл PPD, наданий виробником принтера, перейдіть до його розташування та виберіть його для використання тут. Коли це буде зроблено, натисніть кнопку **Add Printer**.
5. На останній сторінці можна встановити параметри за замовчуванням, наприклад розмір сторінки, який використовуватиме принтер, і роздільну здатність символів, що друкуються на сторінці. Натисніть кнопку **Set Default Options**, і ваш принтер буде встановлено у вашій системі.

NOTE

Багато настільних дистрибутивів Linux матимуть різні інструменти, які можна використовувати для встановлення принтера. Робочі середовища GNOME та KDE мають власні вбудовані програми, які можна використовувати для встановлення та керування принтерами. Крім того, деякі дистрибутиви надають окремі програми для керування принтером. Однак, коли мова йде про встановлення сервера, на якому багато користувачів друкуватимуть, веб-інтерфейс CUPS може надати найкращі інструменти для вирішення цього завдання.

Принтер також можна встановити за допомогою застарілих команд LPD/LPR. Ось приклад використання команди `lpadmin`:

```
$ sudo lpadmin -p ENVY-4510 -L "office" -v socket://192.168.150.25 -m everywhere
```

Ми розберемо команду, щоб проілюструвати параметри, які тут використовуються:

- Оскільки для додавання принтера до системи потрібен користувач із правами адміністратора, перед командою `lpadmin` ми додаємо `sudo`.
- Параметр `-p` є призначенням для ваших завдань друку. Це, по суті, зрозуміле ім'я, щоб користувач міг знати, куди надсилаються завдання друку. Зазвичай ви можете вказати назву принтера.
- Параметр `-L` вказує на розташування принтера. Це необов'язково, але корисно, якщо вам потрібно керувати кількома принтерами в різних місцях.
- Опція `-v` призначена для URI пристрою принтера. URI пристрою – це те, що потрібно

черзі друку CUPS для надсилання відтворених завдань друку на певний принтер. У нашому прикладі ми використовуємо мережеве розташування за допомогою наданої IP-адреси.

- Останній параметр, `-m`, має значення “everywhere”. Він встановлює модель принтера для CUPS, щоб визначити, який файл PPD використовувати. У сучасних версіях CUPS найкраще використовувати “everywhere”, щоб CUPS міг перевірити URI пристрою (встановлений за допомогою попереднього параметра `-v`) і автоматично визначити правильний файл PPD для використання для принтера. У сучасних ситуаціях CUPS використовуватиме лише IPP, як описано нижче.

Як зазначалося раніше, найкраще дозволити CUPS автоматично визначати, який файл PPD використовувати для певної черги друку. Однак застарілу команду `lpinfo` можна використовувати для запиту локально встановлених файлів PPD, щоб побачити, які доступні. Просто вкажіть опцію `--make-and-model` для принтера, який ви бажаєте встановити, і опцію `-m`:

```
$ lpinfo --make-and-model "HP Envy 4510" -m
hplip:0/ppd/hplip/HP/hp-envy_4510_series-hpijs.ppd HP Envy 4510 Series hpijs, 3.17.10
hplip:1/ppd/hplip/HP/hp-envy_4510_series-hpijs.ppd HP Envy 4510 Series hpijs, 3.17.10
hplip:2/ppd/hplip/HP/hp-envy_4510_series-hpijs.ppd HP Envy 4510 Series hpijs, 3.17.10
drv:///hpcups.crv/hp-envy_4510_series.ppd HP Envy 4510 Series, hpcups 3.17.10
everywhere IPP Everywhere
```

Зауважте, що команда `lpinfo` застаріла. Тут показано як приклад перелік файлів драйверів друку, які може використовувати принтер.

WARNING

У майбутніх версіях CUPS драйвери вважатимуться застарілими, і натомість вони будуть зосереджені на використанні IPP (*Internet Printing Protocol*) і стандартних форматів файлів. Результат попередньої команди ілюструє це за допомогою можливості друку `everywhere IPP Everywhere`. IPP може виконувати ті самі завдання, що й драйвер принтера. IPP, як і веб-інтерфейс CUPS, використовує мережевий порт 631 із протоколом TCP.

Принтер за замовчуванням можна встановити за допомогою команди `lpoptions`. Таким чином, якщо більшість (або всі) завдання друку буде надіслано на певний принтер, то принтер, указаний у команді `lpoptions`, буде використовуватись за замовчуванням. Просто вкажіть принтер разом із опцією `-d`:

```
$ lpoptions -d ENVY-4510
```

Керування принтерами

Після встановлення принтера адміністратор може використовувати веб-інтерфейс для керування опціями, доступними для принтера. Більш прямий підхід до керування принтером полягає у використанні команди `lpadmin`.

Одним із варіантів є надання спільного доступу до принтера в мережі. Цього можна досягти за допомогою опції `printer-is-shared` і вказавши принтер за допомогою опції `-p`:

```
$ sudo lpadmin -p FRONT-DESK -o printer-is-shared=true
```

Адміністратор також може налаштувати чергу друку, щоб приймати завдання друку лише від певних користувачів, відокремлюючи кожного користувача комою:

```
$ sudo lpadmin -p FRONT-DESK -u allow:carol,frank,grace
```

І навпаки, доступ до певної черги друку може бути заборонено лише певним користувачам:

```
$ sudo lpadmin -p FRONT-DESK -u deny:dave
```

Групи користувачів також можна використовувати для дозволу або заборони доступу до черги принтера за умови, що перед назвою групи стоїть символ “at” (`@`):

```
$ sudo lpadmin -p FRONT-DESK -u deny:@sales,@marketing
```

Черга друку також може мати політику помилок, якщо під час друку завдання виникають проблеми. За допомогою політик завдання друку може бути перервано (`abort-job`) або інша спроба його друку може відбутися пізніше (`retry-job`). Інші політики включають можливість негайно зупинити принтер у разі виникнення помилки (`stop-printer`), а також можливість повторити завдання одразу після виявлення збою (`retry-current-job`). Ось приклад, коли політика принтера налаштована на переривання завдання друку у разі виникнення помилки на принтері `FRONT-DESK`:

```
$ sudo lpadmin -p FRONT-DESK -o printer-error-policy=abort-job
```

Перегляньте man-сторінки для команди `lpadmin`, розташовані в `lpadmin(8)`, щоб дізнатися більше про використання цієї команди.

Надсилання завдань друку

Багато десктопних програм дозволять вам надсилати завдання друку з пункту меню або за допомогою комбінації клавіш `Ctrl` + `p`. Якщо ви опинитеся в системі Linux, яка не використовує робоче середовище, ви все одно можете надсилати файли на принтер за допомогою застарілих команд LPD/LPR.

Команда `lpr` (“line printer remote”) використовується для надсилання завдання друку до черги принтера. У найпростішій формі команди все, що потрібно: ім’я файлу разом із командою `lpr`:

```
$ lpr report.txt
```

Наведена вище команда надішле файл `report.txt` до черги друку за замовчуванням для системи (як визначено файлом `/etc/cups/printers.conf`).

Якщо CUPS має кілька встановлених принтерів, тоді можна використати команду `lpstat`, щоб роздрукувати список доступних принтерів за допомогою параметра `-p`, а параметр `-d` вкаже, який принтер є типовим:

```
$ lpstat -p -d
printer FRONT-DESK is idle.  enabled since Mon 03 Aug 2020 10:33:07 AM EDT
printer PostScript_oc0303387803 disabled since Sat 07 Mar 2020 08:33:11 PM EST -
    reason unknown
printer ENVY-4510 is idle.  enabled since Fri 31 Jul 2020 10:08:31 AM EDT
system default destination: ENVY-4510
```

Отже, у нашому прикладі файл `report.txt` буде надіслано на принтер `ENVY-4510`, оскільки той встановлений за замовчуванням. Якщо файл потрібно надрукувати на іншому принтері, вкажіть принтер разом із параметром `-P`:

```
$ lpr -P FRONT-DESK report.txt
```

Коли завдання друку надсилається до CUPS, демон визначить, яка серверна частина найкраще підходить для виконання завдання. CUPS може використовувати різноманітні драйвери принтера, фільтри, монітори апаратних портів та інше програмне забезпечення для належного відтворення документа. Траплятимуться випадки, коли користувачеві, який друкує документ, потрібно буде змінити *як* документ має бути надрукований. Багато графічних програм роблять це завдання досить легким. Існують також параметри командного рядка, які можна використовувати для зміни способу друку документа. Коли завдання друку надсилається через командний рядок, перемикач `-o` (для "options") можна використовувати разом із певними термінами, щоб налаштувати макет документа для друку. Ось короткий перелік параметрів, які часто використовуються:

`landscape`

Документ з поворотом сторінки на 90 градусів за годинниковою стрілкою. Параметр `orientation-requested=4` досягне того самого результату.

`two-sided-long-edge`

Принтер друкуватиме документ у портретному режимі на обох сторонах аркуша, за умови, що принтер підтримує цю можливість.

`two-sided-short-edge`

Принтер друкуватиме документ в альбомному режимі на обох сторонах паперу, якщо принтер підтримує цю можливість.

`media`

Принтер роздрукує завдання на вказаному розмірі носія. Розміри носіїв, доступні для завдання друку, залежать від принтера, але ось список поширених розмірів:

Варіант розміру	**Призначення**
`A4`	ISO A4
`Letter`	US Letter
`Legal`	US Legal
`DL`	ISO DL Envelope
`COM10`	US #10 Envelope

`collate`

Сортувати друкований документ. Це корисно, якщо у вас є багатосторінковий документ, який буде надруковано кілька разів, оскільки тоді всі сторінки кожного документа друкуватимуться по порядку. Встановіть для цього параметра значення `true`, щоб

увімкнути його, або `false`, щоб вимкнути його.

`page-ranges`

Цей параметр можна використовувати для вибору однієї сторінки для друку або певного набору сторінок для друку з документа. Приклад виглядатиме так: `-o page-ranges=5-7,9,15`. Буде надруковано сторінки 5, 6 і 7, а потім сторінки 9 і 15.

`fit-to-page`

Роздрукувати документ так, щоб файл був масштабований відповідно до паперу. Якщо файл, який потрібно надрукувати, не містить інформації про розмір сторінки, можливо, друковане завдання буде масштабовано неправильно, і частки документа можуть бути за межами сторінки, або масштаб документа може бути занадто малим.

`outputorder`

Надрукуйте документ у `зворотному (reverse)` порядку або `звичайному (normal)`, щоб розпочати друк із першої сторінки. Якщо принтер друкує свої сторінки лицьовою стороною донизу, за замовчуванням порядок `-o outputorder=normal`, тоді як принтери, які друкують сторінками лицьовою стороною догори, друкуватимуть з `-o outputorder=reverse`.

Зробивши комбінацію з параметрів вище, можна сконструювати такий приклад команди:

```
$ lpr -P ACCOUNTING-LASERJET -o landscape -o media=A4 -o two-sided-short-edge finance-
report.pdf
```

Більш ніж одну копію документа можна надрукувати за допомогою опції числа в такому форматі: `-#N`, де `N` дорівнює кількості копій для друку. Ось приклад із параметром зіставлення, коли сім копій звіту потрібно надрукувати на принтері за замовчуванням:

```
$ lpr -#7 -o collate=true status-report.pdf
```

Окрім команди `lpr`, також можна використовувати команду `lp`. Багато параметрів, які використовуються з командою `lpr`, також можна використовувати з командою `lp`, але є деякі відмінності. Перегляньте man-сторінку для `lp(1)`. Ось як ми можемо запустити попередній приклад команди `lpr`, використовуючи синтаксис команди `lp`, одночасно вказуючи принтер призначення за допомогою параметра `-d`:

```
$ lp -d ACCOUNTING-LASERJET -n 7 -o collate=true status-report.pdf
```

Керування завданнями друку

Як було зазначено раніше, кожне завдання друку, надіслане до черги друку, отримує ідентифікатор завдання від CUPS. Користувач може переглядати завдання друку, надіслані за допомогою команди `lpq`. Передача параметра `-a` покаже черги всіх принтерів, якими керує CUPS:

```
$ lpq -a
Rank        Owner         Job       File(s)                 Total Size
1st         carol         20        finance-report.pdf      5072 bytes
```

Та сама команда `lpstat`, яка використовувалася раніше, також має опцію для перегляду черг принтера. Параметр `-o` сам по собі покаже всі черги друку, або чергу друку можна вказати за назвою:

```
$ lp -o
ACCOUNTING-LASERJET-4                  carol        19456   Wed 05 Aug 2020 04:29:44 PM EDT
```

Перед ідентифікатором завдання друку буде додано назву черги, куди було надіслано завдання, потім ім'я користувача, який надіслав завдання, розмір файлу та час його надсилання.

Якщо завдання друку застрягло на принтері або користувач бажає скасувати своє завдання друку, скористайтеся командою `lprm` разом із ідентифікатором завдання, знайденим командою `lpq`:

```
$ lprm 20
```

Усі завдання в черзі друку можна видалити відразу, поставивши лише дефіс `-`:

```
$ lprm -
```

Крім того, користувач також може використати команду CUPS `cancel` для зупинки поточного завдання друку:

```
$ cancel
```

Певне завдання друку можна скасувати за ідентифікатором завдання, перед яким стоїть

назва принтера:

```
$ cancel ACCOUNTING-LASERJET-20
```

Завдання друку також можна перемістити з однієї черги друку до іншої. Це часто корисно, якщо принтер не відповідає або для друку документа потрібні функції, доступні на іншому принтері. Зверніть увагу, що ця процедура зазвичай потребує користувача з підвищеними правами. Використовуючи те саме завдання друку з попереднього прикладу, ми можемо перемістити його до черги принтера `FRONT-DESK`:

```
$ sudo lpmove ACCOUNTING-LASERJET-20 FRONT-DESK
```

Видалення принтерів

Щоб видалити принтер, часто корисно спочатку вивести список усіх принтерів, якими наразі керує служба CUPS. Це можна зробити за допомогою команди `lpstat`:

```
$ lpstat -v
device for FRONT-DESK: socket://192.168.150.24
device for ENVY-4510: socket://192.168.150.25
device for PostScript_oc0303387803: ///dev/null
```

Параметр `-v` не лише показує список принтерів, але й те, де (і як) вони підключені. Рекомендується спочатку відхилити будь-які нові завдання, які надходять на принтер, і вказати причину, чому принтер не прийматиме нові завдання. Це можна зробити наступним чином:

```
$ sudo cupsreject -r "Printer to be removed" FRONT-DESK
```

Зверніть увагу на використання `sudo`, оскільки це завдання потребує користувача з підвищеними правами.

Щоб видалити принтер, ми використовуємо команду `lpadmin` з опцією `-x` для видалення принтера:

```
$ sudo lpadmin -x FRONT-DESK
```

Вправи до посібника

1. Новий принтер щойно встановлено на локальній робочій станції під назвою `office-mgr`. Яку команду можна використати, щоб зробити його принтером за замовчуванням для цієї робочої станції?

2. Яку команду та параметр використовуватимуть, щоб визначити, які принтери доступні для друку з робочої станції?

3. Як би ви видалили завдання друку з ідентифікатором 15, яке застрягло в черзі для принтера під назвою `office-mgr`, за допомогою команди `cancel`?

4. У вас є завдання друку, призначене для принтера, на якому недостатньо паперу для друку всього файлу. Яку команду ви використаєте, щоб перемістити завдання друку з ID 2 у черзі для друку на принтері `FRONT-DESK` до черги друку принтера `ACCOUNTING-LASERJET`?

Дослідницькі вправи

За допомогою менеджера пакунків дистрибутива встановіть пакунки `cups` і `printer-driver-cups-pdf`. Зауважте, що якщо ви використовуєте дистрибутив на основі Red Hat (наприклад, Fedora), драйвер CUPS PDF називається `cups-pdf`. Також встановіть пакет `cups-client`, щоб використовувати команди друку в стилі System V. Ми використовуватимемо ці пакунки, щоб практикувати керування принтером CUPS без фізичного встановлення справжнього принтера.

1. Переконайтеся, що демон CUPS запущено, а потім переконайтеся, що PDF-принтер увімкнено та встановлено за замовчуванням.

2. Виконайте команду, яка роздрукує файл `/etc/services`. Тепер у вашому домашньому каталозі має бути каталог під назвою `PDF`.

3. Використовуйте команду, яка вимкне лише принтер, а потім запустіть окрему команду, яка показує всю інформацію про стан, щоб переконатися, що PDF-принтер вимкнено. Потім спробуйте надрукувати копію вашого файлу `/etc/fstab`. Що станеться?

4. Тепер спробуйте надрукувати копію файлу `/etc/fstab` на PDF-принтері. Що станеться?

5. Скасуйте завдання друку, а потім видаліть PDF-принтер.

Підсумки

Демон CUPS — це платформа для друку на локальних і віддалених принтерах, що широко використовується. Хоча він замінює застарілий протокол LPD, він усе ще забезпечує зворотну сумісність для його інструментів.

У цьому уроці розглядалися такі файли та команди:

`/etc/cups/cupsd.conf`

Основний файл конфігурації для самої служби CUPS. Цей файл також керує доступом до веб-інтерфейсу для CUPS.

`/etc/printcap`

Застарілий файл, який використовується LPD і містить рядок для кожного принтера, підключеного до системи.

`/etc/cups/printers.conf`

Файл конфігурації, який використовується CUPS для інформації про принтер.

Веб-інтерфейс CUPS, який за умовчанням можна знайти за адресою `http://localhost:631`. Пам'ятайте, що стандартним мережевим портом для веб-інтерфейсу є 631/TCP.

Також обговорювалися наступні застарілі команди LPD/LPR:

`lpadmin`

Використовується для встановлення та видалення принтерів і класів принтерів.

`lpoptions`

Використовується для друку параметрів принтера та зміни налаштувань принтера.

`lpstat`

Використовується для відображення інформації про стан принтерів, які підключені до CUPS-інсталяції.

`lpr`

Використовується для надсилання завдань друку до черги принтера.

`lp`

Використовується для надсилання завдань друку до черги принтера.

`lpq`

Ця команда показує список завдань друку в черзі друку.

`lprm`

Використовується для скасування завдань друку за ідентифікатором. Ідентифікатор завдання можна отримати за допомогою виводу команди `lpq`.

`cancel`

Альтернатива команді `lprm` для скасування завдань друку за їхнім ідентифікатором.

Перегляньте наступні довідкові сторінки для різних інструментів і утиліт для cups: `lpadmin(8)`, `lpoptions(1)`, `lpr(1)`, `lpq(1)`, `lprm(1)`, `cancel(1)`, `lpstat(1)`, `cupsenable(8)` і `cupsaccept(8)`. Рекомендується також переглянути онлайн-довідкову документацію за адресою `http://localhost:631/help`.

Відповіді до вправ посібника

1. Новий принтер щойно встановлено на локальній робочій станції під назвою `office-mgr`. Яку команду можна використати, щоб зробити його принтером за замовчуванням для цієї робочої станції?

   ```
   $ lpoptions -d office-mgr
   ```

2. Яку команду та параметр використовуватимуть, щоб визначити, які принтери доступні для друку з робочої станції?

   ```
   $ lpstat -p
   ```

   Параметр `-p` показує список усіх доступних принтерів і чи їх увімкнено для друку.

3. Як би ви видалили завдання друку з ідентифікатором 15, яке застрягло в черзі для принтера під назвою `office-mgr`, за допомогою команди `cancel`?

   ```
   $ cancel office-mgr-15
   ```

4. У вас є завдання друку, призначене для принтера, на якому недостатньо паперу для друку всього файлу. Яку команду ви використаєте, щоб перемістити завдання друку з ID 2 у черзі для друку на принтері `FRONT-DESK` до черги друку принтера `ACCOUNTING-LASERJET`?

   ```
   $ sudo lpmove FRONT-DESK-2 ACCOUNTING-LASERJET
   ```

Відповіді до дослідницьких вправ

За допомогою менеджера пакунків дистрибутива встановіть пакунки `cups` і `printer-driver-cups-pdf`. Зауважте, що якщо ви використовуєте дистрибутив на основі Red Hat (наприклад, Fedora), драйвер CUPS PDF називається `cups-pdf`. Також встановіть пакет `cups-client`, щоб використовувати команди друку в стилі System V. Ми використовуватимемо ці пакунки, щоб практикувати керування принтером CUPS без фізичного встановлення справжнього принтера.

1. Переконайтеся, що демон CUPS запущено, а потім переконайтеся, що PDF-принтер увімкнено та встановлено за замовчуванням.

   Одним із способів перевірити доступність і стан PDF-принтера є виконання такої команди:

   ```
   $ lpstat -p -d

   printer PDF is idle.  enabled since Thu 25 Jun 2020 02:36:07 PM EDTi

   system default destination: PDF
   ```

2. Виконайте команду, яка роздрукує файл `/etc/services`. Тепер у вашому домашньому каталозі має бути каталог під назвою `PDF`.

   ```
   $ lp -d PDF /etc/services
   ```

   спрацювала б. Тепер ви матимете PDF-версію цього файлу в каталозі PDF.

3. Використовуйте команду, яка вимкне лише принтер, а потім запустіть окрему команду, яка показує всю інформацію про стан, щоб переконатися, що PDF-принтер вимкнено.

   ```
   $ sudo cupsdisable PDF
   ```

   вимкне принтер.

   Потім запустіть команду `lpstat -t`, щоб отримати повну інформацію про стан принтера. Це має виглядати наступним чином:

   ```
   $ scheduler is running
   ```

```
system default destination: PDFi

device for PDF: cups-pdf:/

PDF accepting requests since Wed 05 Aug 2020 04:19:15 PM EDTi

printer PDF disabled since Wed 05 Aug 2020 04:19:15 PM EDT -

Paused
```

4. Тепер спробуйте надрукувати копію файлу `/etc/fstab` на PDF-принтері. Що станеться?

   Після спроби виконати команду `lp -d PDF /etc/fstab` ви маєте отримати вихідні дані з ідентифікатором завдання. Однак якщо ви перевірите папку PDF у своєму домашньому каталозі, нового файлу там немає. Потім ви можете перевірити чергу друку за допомогою команди `lpstat -o` і ви знайдете там своє завдання.

5. Скасуйте завдання друку, а потім видаліть PDF-принтер.

   Використовуючи результат попередньої команди `lp`, використовуйте команду `cancel`, щоб видалити завдання. Наприклад:

```
$ cancel PDF-4
```

   Потім запустіть команду `lpstat -o`, щоб переконатися, що завдання видалено.

   Видаліть PDF-принтер за допомогою такої команди: `sudo lpadmin -x PDF`. Потім переконайтеся, що принтер видалено: `lpstat -a`.

Linux
Professional
Institute

Розділ 109: Основи комп’ютерних мереж

109.1 Основи інтернет-протоколів

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 109.1

Ваговий коефіцієнт

4

Ключові галузі знань

- Демонстрація розуміння мережних масок і нотації CIDR.
- Знання відмінностей між приватними та загальнодоступними IP-адресами.
- Знання про загальні порти та служби TCP і UDP (20, 21, 22, 23, 25, 53, 80, 110, 123, 139, 143, 161, 162, 389, 443, 465, 514, 636, 993, 995) .
- Знання про відмінності та основні особливості UDP, TCP та ICMP.
- Знання основних відмінностей між IPv4 і IPv6.
- Знання основних функцій IPv6.

Частковий список файлів, термінів та утиліт, що використовуються

- `/etc/services`
- IPv4, IPv6
- Підмережі
- TCP, UDP, ICMP

109.1 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	109 Основи комп’ютерних мереж
Тема:	109.1 Основи інтернет-протоколів
Урок:	1 з 2

Вступ

TCP/IP (*Протокол керування передачею/Інтернет-протокол*) — це стек протоколів, які використовуються для забезпечення зв’язку між комп’ютерами. Незважаючи на назву, стек складається з кількох протоколів, таких як IP, TCP, UDP, ICMP, DNS, SMTP, ARP та інші.

IP (Інтернет протокол)

IP — це протокол, відповідальний за логічну адресацію хоста, що дозволяє надсилати пакети з одного хоста на інший. Для цього кожному пристрою в мережі призначається унікальна IP-адреса, за необхідністю одному пристрою можна призначити більше однієї адреси.

У версії 4 протоколу IP, яка зазвичай називається IPv4, адреса формується набором із 32 бітів, розділених на 4 групи по 8 бітів, представлених у десятковій формі, що називається “dotted quad”. Наприклад:

Двійковий формат (4 групи по 8 біт)

```
11000000.10101000.00001010.00010100
```

Десятковий формат

`192.168.10.20`

У IPv4 значення для кожного октету можуть варіюватися від 0 до 255, що є еквівалентом 11111111 у двійковому форматі.

Класова адресація

Теоретично IP-адреси розділяються за класами, які визначаються діапазоном першого октету, як показано в таблиці нижче:

Клас	Перший октект	Діапазон	приклад
A	1-126	`1.0.0.0` – `126.255.255.255`	`10.25.13.10`
B	128-191	`128.0.0.0` – `191.255.255.255`	`141.150.200.1`
C	192-223	`192.0.0.0` – `223.255.255.255`	`200.178.12.242`

Публічні та приватні IP-адреси

Як згадувалося раніше, для встановлення зв'язку кожен пристрій у мережі має бути пов'язаний принаймні з однією унікальною IP-адресою. Однак, якби кожен пристрій у світі, підключений до Інтернету, мав унікальну IP-адресу, не вистачило б IP-адрес (v4) для всіх. Для цього були визначені *приватні* IP-адреси.

Приватні IP-адреси – це діапазони IP-адрес, які були зарезервовані для використання у внутрішніх (приватних) мережах компаній, установ, будинків тощо. У межах однієї мережі використання IP-адреси залишається унікальним. Однак ту саму приватну IP-адресу можна використовувати в будь-якій приватній мережі.

Таким чином, в Інтернеті ми маємо трафік даних з використанням загальнодоступних IP-адрес, які розпізнаються та маршрутизуються через Інтернет, тоді як у приватних мережах використовуються ці зарезервовані IP-діапазони. Маршрутизатор відповідає за перетворення трафіку з приватної мережі в публічну мережу і навпаки.

Діапазони приватних IP-адрес, розділені за класами, можна побачити в таблиці нижче:

Клас	Перший октет	Діапазон	Приватні IP-адреси
A	1-126	`1.0.0.0` – `126.255.255.255`	`10.0.0.0` – `10.255.255.255`
B	128-191	`128.0.0.0` – `191.255.255.255`	`172.16.0.0` – `172.31.255.255`
C	192-223	`192.0.0.0` – `223.255.255.255`	`192.168.0.0` – `192.168.255.255`

Перетворення з десяткового формату на двійковий

Для цієї теми важливо знати, як конвертувати IP-адреси між двійковим і десятковим форматами.

Перетворення з десяткового формату на двійковий виконується шляхом послідовних ділень на 2. Для прикладу давайте перетворимо значення 105 за допомогою таких кроків:

1. Розділивши значення 105 на 2, отримаємо:

```
105/2
Частка = 52
Остача = 1
```

2. Частку послідовно ділимо на 2, поки частка не дорівнюватиме 1:

```
52/2
Остача = 0
Частка = 26
```

```
26/2
Остача = 0
Частка = 13
```

```
13/2
Остача = 1
Частка = 6
```

```
6/2
```

```
Остача = 0
Частка = 3
```

```
3/2
Остача = 1
Частка = 1
```

3. Згрупуйте останню частку, а потім залишок від усіх ділень:

```
1101001
```

4. Додайте 0 зліва, доки не буде заповнено 8 бітів:

```
01101001
```

5. Зрештою, ми маємо, що значення 105 у десятковій системі дорівнює 01101001 у двійковій системі.

Перетворення з двійкового формату в десятковий

У цьому прикладі ми будемо використовувати двійкове значення `10110000`.

1. Кожен біт пов'язаний зі значенням двійки. Ступені починаються з 0 і збільшуються справа наліво. У цьому прикладі ми матимемо:

1	0	1	1	0	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

2. Коли біт дорівнює 1, ми призначаємо значення відповідного ступеня, коли біт дорівнює 0, результат дорівнює 0.

1	0	1	1	0	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	0	32	16	0	0	0	0

3. Складіть усі значення:

   128 + 32 + 16 = 176

4. Таким чином, 10110000 у двійковій системі дорівнює 176 у десятковій системі.

Мережева маска

Маска мережі (або *netmask*) використовується в поєднанні з IP-адресою, щоб визначити, яка частина IP-адреси представляє мережу, а яка — хости. Вона має той самий формат, що й IP-адреса, тобто є 32 біти в 4 групах по 8 в кожній. Наприклад:

Десятковий	**Двійковий**	**CIDR**
`255.0.0.0`	`11111111.00000000.00000000.00000000`	`8`
`255.255.0.0`	`11111111.11111111.00000000.00000000`	`16`
`255.255.255.0`	`11111111.11111111.11111111.00000000`	`24`

Використовуючи маску `255.255.0.0` як приклад, маємо в пов'язаній з нею IP-адресі, що перші 16 бітів (2 перші десяткові знаки) ідентифікують мережу/підмережу, а останні 16 бітів використовуються для унікальної ідентифікації хостів у межах мережі.

CIDR (*Безкласова міждоменна маршрутизація*), згадана вище, пов'язана зі спрощеною нотацією маски, яка вказує на кількість бітів (1), пов'язаних з мережею/підмережею. Ця нотація зазвичай використовується для заміни десяткового формату, наприклад, `/24` замість `255.255.255.0`.

Цікаво відзначити, що кожен клас IP має стандартну маску, а саме:

Клас	**Перший октет**	**Діапазон**	**Типова маска**
A	1-126	`1.0.0.0` – `126.255.255.255`	`255.0.0.0 / 8`
B	128-191	`128.0.0.0` – `191.255.255.255`	`255.255.0.0 / 16`
C	192-223	`192.0.0.0` – `223.255.255.255`	`255.255.255.0 / 24`

Однак цей шаблон не означає, що це та маска, яка буде використовуватися завжди. Можна використовувати будь-яку маску з будь-якою IP-адресою, як ми побачимо нижче.

Ось кілька прикладів використання IP-адрес і масок:

```
192.168.8.12 / 255.255.255.0 / 24
```

Діапазон

`192.168.8.0` - `192.168.8.255`

Адреса мережі

`192.168.8.0`

Широкомовна адреса

`192.168.8.255`

Хости

`192.168.8.1` - `192.168.8.254`

У цьому випадку перші 3 числа (перші 24 біти) IP-адреси визначають мережу, а останнє число ідентифікує адреси хостів, тобто діапазон цієї мережі змінюється від `192.168.8.0` до `192.168.8.255`.

Тепер у нас є дві важливі концепції: кожна мережа/підмережа має 2 зарезервовані адреси, перша адреса в діапазоні називається *мережева адреса*. У цьому випадку адреса `192.168.8.0` використовується для ідентифікації самої мережі/підмережі. Остання адреса в діапазоні називається *широкомовною адресою*, у цьому випадку `192.168.8.255`. Ця адреса призначення використовується для надсилання повідомлення (пакета) усім IP-хостам у цій мережі/підмережі.

Мережні та широкомовні адреси не можуть використовуватися машинами в мережі. Тому список IP-адрес, які можна ефективно налаштувати, коливається від `192.168.8.1` до `192.168.8.254`.

Тепер приклад того ж IP, але з іншою маскою:

```
192.168.8.12 / 255.255.0.0 / 16
```

Діапазон

`192.168.0.0` - `192.168.255.255`

Адреса мережі

`192.168.0.0`

Широкомовна адреса

`192.168.255.255`

Хости

`192.168.0.1` – `192.168.255.254`

Подивіться, як інша маска змінює діапазон IP-адрес, які знаходяться в одній мережі/підмережі.

Поділ мереж за масками не обмежений значеннями за замовчуванням (8, 16, 24). Ми можемо створювати поділи за бажанням, додаючи або видаляючи біти в ідентифікації мережі, створюючи нові підмережі. с Наприклад:

```
11111111.11111111.11111111.00000000 = 255.255.255.0 = 24
```

Якщо ми хочемо розділити мережу вище на 2, просто додайте ще один біт до ідентифікації мережі в масці, ось так:

```
11111111.11111111.11111111.10000000 = 255.255.255.128 = 25
```

Тоді ми маємо такі підмережі:

```
192.168.8.0   - 192.168.8.127
192.168.8.128 - 192.168.8.255
```

Якщо ми ще збільшимо поділ мережі:

```
11111111.11111111.11111111.11000000 = 255.255.255.192 = 26
```

Ми отримаємо:

```
192.168.8.0   - 192.168.8.63
192.168.8.64  - 192.168.8.127
192.168.8.128 - 192.168.8.191
192.168.8.192 - 192.168.8.255
```

Зауважте, що в кожній підмережі ми матимемо зарезервовану мережеву (першу в діапазоні) і широкомовну (останню в діапазоні) адреси, тому чим більше мережа поділена,

тим менше IP-адрес можуть ефективно використовуватися хостами.

Визначення мережі та широкомовних адрес

За допомогою IP-адреси та маски ми можемо ідентифікувати мережеву адресу та широкомовну адресу та, таким чином, визначити діапазон IP-адрес для мережі/підмережі.

Мережева адреса отримується за допомогою "логічного ТА" між IP-адресою та маскою в їх двійкових форматах. Розглянемо приклад із IP-адресою `192.168.8.12` і маскою `255.255.255.192`.

Перетворюючи з десяткового у двійковий формат, як ми бачили раніше, ми маємо:

```
11000000.10101000.00001000.00001100 (192.168.8.12)
11111111.11111111.11111111.11000000 (255.255.255.192)
```

За допомогою "логічного ТА" ми маємо 1 ТА 1 = 1, 0 ТА 0 = 0, 1 ТА 0 = 0, отже:

```
11000000.10101000.00001000.00001100 (192.168.8.12)
11111111.11111111.11111111.11000000 (255.255.255.192)
11000000.10101000.00001000.00000000
```

Отже, мережна адреса для цієї підмережі `192.168.8.0`.

Тепер, щоб отримати широкомовну адресу, ми повинні використовувати мережну адресу, де всі біти, пов'язані з адресою хоста, дорівнюють 1:

```
11000000.10101000.00001000.00000000 (192.168.8.0)
11111111.11111111.11111111.11000000 (255.255.255.192)
11000000.10101000.00001000.00111111
```

Тоді широкомовна адреса буде `192.168.8.63`.

В результаті маємо:

```
192.168.8.12 / 255.255.255.192 / 26
```

Діапазон

`192.168.8.0` - `192.168.8.63`

Адреса мережі

`192.168.8.0`

Широкомовна адреса

`192.168.8.63`

Хости

`192.168.8.1` – `192.168.8.62`

Маршрут за умовчанням

Як ми вже бачили, машини, які знаходяться в одній логічній мережі/підмережі, можуть спілкуватися безпосередньо через протокол IP.

Але давайте розглянемо приклад нижче:

Мережа 1

`192.168.10.0/24`

Мережа 2

`192.168.200.0/24`

У цьому випадку машина 192.168.10.20 не може безпосередньо надіслати пакет на 192.168.200.100, оскільки вони знаходяться в різних логічних мережах.

Для забезпечення такого зв'язку використовується маршрутизатор (або набір маршрутизаторів). Маршрутизатор у цій конфігурації також можна назвати шлюзом, оскільки він забезпечує шлюз між двома мережами. Цей пристрій має доступ до обох мереж, оскільки в ньому налаштовано IP-адреси з обох мереж. Наприклад, `192.168.10.1` і `192.168.200.1`, і з цієї причини йому вдається бути посередником у такому спілкуванні.

Для комунікації між мережами кожен хост у мережі має налаштувати так званий *маршрут за замовчуванням*. Маршрут за замовчуванням вказує IP-адресу, на яку мають надсилатися всі пакети, адресатом яких є IP-адреса, що не є частиною логічної мережі хоста.

У наведеному вище прикладі маршрутом за замовчуванням для машин у мережі `192.168.10.0/24` буде IP-адреса `192.168.10.1`, яка є IP-адресою маршрутизатора/шлюзу, а маршрутом за замовчуванням для машин у мережі `192.168.200.0/24` буде `192.168.200.1`.

Маршрут за замовчуванням також використовується, щоб машини в приватній мережі

(LAN) мали доступ до Інтернету (WAN) через маршрутизатор.

Вправи до посібника

1. Використовуючи IP-адресу `172.16.30.230` і маску мережі `255.255.255.224`, визначте:

Нотація CIDR для маски мережі	
Адреса мережі	
Широкомовна адреса	
Кількість IP-адрес, які можна використовувати для хостів у цій підмережі	

2. Яке налаштування потрібно на хості, щоб дозволити IP-з'єднання з хостом в іншій логічній мережі?

Дослідницькі вправи

1. Чому діапазони IP-адрес, що починаються зі `127` і діапазон після `224`, не входять до класів IP-адрес A, B або C?

2. Одним із дуже важливих полів IP-пакета є TTL (*Time To Live*). Яку функцію виконує це поле і як воно працює?

3. Поясніть функцію NAT і навіщо вона потрібна.

Підсумки

У цьому уроці розглядалися основні концепції протоколу IPv4, який відповідає за забезпечення зв’язку між хостами в мережі.

Також були вивчені основні операції, які професіонал повинен знати, щоб конвертувати IP-адреси в різні формати, а також мати можливість аналізувати та виконувати логічні конфігурації в мережах і підмережах.

Були розглянуті наступні теми:

- Класи IP-адрес
- Публічні та приватні IP-адреси
- Як конвертувати IP-адреси з десяткового у двійковий формат і навпаки
- Маска мережі (netmask)
- Як ідентифікувати мережу та широкомовні адреси за IP та маскою мережі
- Маршрут за замовчуванням

Відповіді до вправ посібника

1. Використовуючи IP-адресу `172.16.30.230` і маску мережі `255.255.255.224`, визначте:

Нотація CIDR для маски мережі	`27`
Адреса мережі	`172.16.30.224`
Широкомовна адреса	`172.16.30.255`
Кількість IP-адрес, які можна використовувати для хостів у цій підмережі	`30`.

2. Яке налаштування потрібно на хості, щоб дозволити IP-з'єднання з хостом в іншій логічній мережі?

   Маршрут за замовчуванням

Відповіді до дослідницьких вправ

1. Чому діапазони IP-адрес, що починаються зі `127` і діапазон після `224`, не входять до класів IP-адрес A, B або C?

   Діапазон, який починається зі `127`, зарезервований для петлевих (loopback) адрес, які використовуються для тестів і внутрішнього зв'язку між процесами, наприклад адреса `127.0.0.1`. Крім того, адреси вище `224` також використовуються не як адреси хостів, а для багатоадресної передачі та інших цілей.

2. Одним із дуже важливих полів IP-пакета є TTL (*Time To Live*). Яку функцію виконує це поле і як воно працює?

   TTL визначає час життя пакета. Це реалізується за допомогою лічильника, у якому початкове значення, визначене в джерелі, зменшується на кожному шлюзі/маршрутизаторі, через який проходить пакет, що також називається "стрибком". Якщо цей лічильник досягає 0, пакет відкидається.

3. Поясніть функцію NAT і навіщо вона потрібна.

   Функція NAT (*Трансляція мережевих адрес*) дозволяє хостам у внутрішній мережі, які використовують приватні IP-адреси, мати доступ до Інтернету так, ніби вони підключені безпосередньо до нього, з публічною IP-адресою, яка використовується на шлюзі.

109.1 Урок 2

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	109 Основи комп'ютерних мереж
Тема:	109.1 Основи інтернет-протоколів
Урок:	2 з 2

Вступ

На початку цього підрозділу ми побачили, що стек TCP/IP складається з низки різних протоколів. Поки що ми вивчали протокол IP, який дозволяє спілкуватися між комп'ютерами через IP-адреси, маски, маршрути тощо.

Щоб хост міг отримати доступ до служби, доступної на іншому хості, на додаток до протоколу IP-адресації на мережевому рівні, необхідно буде використовувати протокол на транспортному рівні, наприклад протоколи TCP і UDP.

Ці протоколи здійснюють зв'язок через мережеві порти. Таким чином, окрім визначення IP-адреси джерела та призначення, також порти джерела та призначення використовуватимуться для доступу до служби.

Порт ідентифікується 16-бітним полем, що забезпечує обмеження в 65 535 можливих портів. Служби (призначення) використовують порти від 1 до 1023, які називаються *привілейованими портами*, оскільки вони мають кореневий доступ до системи. Джерело підключення буде використовувати діапазон портів від 1024 до 65 535, які називаються *непривілейованими портами* або портами сокетів.

Порти, які використовуються кожним типом служби, стандартизовані та контролюються IANA (*Internet Assigned Numbers Authority*). Це означає, що в будь-якій системі порт 22 використовується службою SSH, порт 80 – службою HTTP тощо.

Таблиця нижче містить основні служби та відповідні порти.

Порт	**Сервіс**
20	FTP (дані)
21	FTP (керування)
22	SSH (Secure Socket Shell)
23	Telnet (Віддалене підключення без шифрування)
25	SMTP (Простий протокол передачі пошти), Надсилання пошти
53	DNS (система доменних імен)
80	HTTP (протокол передачі гіпертексту)
110	POP3 (протокол поштового відділення, версія 3), отримання листів
123	NTP (протокол мережевого часу)
139	Netbios
143	IMAP (Internet Message Access Protocol), доступ до пошти
161	SNMP (простий протокол керування мережею)
162	SNMPTRAP, сповіщення SNMP
389	LDAP (полегшений протокол доступу до каталогу)
443	HTTPS (захищений HTTP)
465	SMTPS (захищений SMTP)
514	RSH (Remote Shell)
636	LDAPS (захищений LDAP)
993	IMAPS (захищений IMAP)
995	POP3S (захищений POP3)

У системі Linux стандартні службові порти перераховані у файлі `/etc/services`.

Ідентифікація потрібного порту призначення в з'єднанні здійснюється за допомогою символу `:` (двокрапка) після адреси IPv4. Таким чином, шукаючи доступ до служби HTTPS, яка обслуговується IP-хостом `200.216.10.15`, клієнт повинен надіслати запит до пункту призначення `200.216.10.15:443`.

Перелічені вище служби та всі інші використовують транспортний протокол відповідно до характеристик, необхідних для служби, де TCP і UDP є основними.

Протокол керування передачею (TCP)

TCP - це транспортний протокол, орієнтований на з'єднання. Це означає, що з'єднання встановлюється між клієнтом через порт сокета та службою через стандартний порт служби. Протокол відповідає за забезпечення належної доставки всіх пакунків, перевірку цілісності та порядку пакунків, включаючи повторну передачу пакунків, втрачених через помилки мережі.

Таким чином, програмі не потрібно впроваджувати цей контроль потоку даних, оскільки це вже гарантовано протоколом TCP.

Протокол дейтаграм користувача (UDP)

UDP встановлює з'єднання між клієнтом і службою, але не контролює передачу даних цього з'єднання. Іншими словами, він не перевіряє, чи не були пакети втрачені, чи були вони пошкоджені тощо. Застосунок несе відповідальність за впровадження необхідних елементів керування.

Через менший контроль UDP забезпечує кращу продуктивність потоку даних, що важливо для деяких типів сервісів.

Міжмережевий протокол керуючих повідомлень (ICMP)

ICMP — це протокол мережевого рівня в стеку TCP/IP, і його основна функція полягає в аналізі та контролі мережевих елементів, що робить можливим, наприклад:

- Контроль гучності трафіку.
- Виявлення недоступних місць призначення.
- Перенаправлення маршруту.
- Перевірку стану віддалених хостів.

Це протокол, який використовується командою `ping`, яка буде розглянута в іншому підрозділі.

IPv6

Поки що ми вивчали версію 4 протоколу IP, тобто IPv4. Це стандартна версія, яка використовується в усіх мережах та Інтернет-середовищах. Однак цей протокол має обмеження, особливо щодо кількості доступних адрес, і з огляду на те, що всі пристрої певним чином підключені до Інтернету (див. IoT), стає все більш поширеним використання версії 6 протоколу IP, зазвичай позначається як IPv6.

IPv6 приносить низку змін, нові реалізації та функції, а також нове представлення самої адреси.

Кожна адреса IPv6 має 128 біт, розділених на 8 груп по 16 біт, представлених шістнадцятковими значеннями.

Наприклад:

```
2001:0db8:85a3:08d3:1319:8a2e:0370:7344
```

Скорочення

IPv6 визначає способи скорочення адрес у деяких ситуаціях. Давайте розглянемо таку адресу:

```
2001:0db8:85a3:0000:0000:0000:0000:7344
```

Перша можливість полягає в тому, щоб скоротити рядки від `0000` до просто `0`, що призведе до:

```
2001:0db8:85a3:0:0:0:0:7344
```

Крім того, у випадку групових рядків зі значенням `0` їх можна опустити, як показано нижче:

```
2001:0db8:85a3::7344
```

Однак останнє скорочення можна ввести в адресі лише один раз. Дивіться приклад:

```
2001:0db8:85a3:0000:0000:1319:0000:7344

2001:0db8:85a3:0:0:1319:0:7344

2001:0db8:85a3::1319:0:7344
```

Типи адрес IPv6

IPv6 класифікує адреси на 3 типи:

Індивідуальна

Ідентифікує єдиний мережевий інтерфейс. За замовчуванням 64 біти ліворуч ідентифікують мережу, а 64 біти праворуч ідентифікують інтерфейс.

Групова

Ідентифікує набір мережевих інтерфейсів. Пакет, надісланий на групову адресу, буде надіслано на всі інтерфейси, які належать до цієї групи. Незважаючи на схожість, її не слід плутати з широкомовною адресою, якої не існує в протоколі IPv6.

Альтернативна

Ця адреса також ідентифікує набір інтерфейсів у мережі, але пакет, що пересилається на *альтернативну* адресу, буде доставлено лише на одну адресу в цьому наборі, а не на всі.

Відмінності між IPv4 і IPv6

Окрім подання адрес, між версіями 4 і 6 IP можна зазначити кілька інших відмінностей. Ось деякі з них:

- Сервісні порти відповідають однаковим стандартам і протоколам (TCP, UDP), різниця лише в представленні IP-адреси та набору портів. У IPv6 IP-адреса має бути заключена у `[]` (дужки):

  IPv4

  `200.216.10.15:443`

  IPv6

  `[2001:0db8:85a3:08d3:1319:8a2e:0370:7344]:443`

- IPv6 не реалізує функцію широкомовної розсилки так, як це існує в IPv4. Однак такого ж

результату можна досягти, надіславши пакет на адресу `ff02::1`, досягаючи всіх хостів у локальній мережі. Щось подібне до використання `224.0.0.1` на IPv4 для багатоадресної розсилки як адреси призначення.

- Завдяки функції SLAAC (*Stateless Address Autoconfiguration*) хости IPv6 можуть самостійно налаштовуватися.
- Поле TTL (*Час життя пакунку*) IPv4 було замінено на “Hop Limit” у заголовку IPv6.
- Усі інтерфейси IPv6 мають локальну адресу, яка називається локальною адресою посилання, із префіксом `fe80::/10`.
- IPv6 реалізує *протокол виявлення сусідів* (NDP), який подібний до ARP, що використовується в IPv4, але з набагато більшою функціональністю.

Вправи до посібника

1. Який порт є стандартним для протоколу SMTP?

2. Скільки різних портів доступно в системі?

3. Який транспортний протокол забезпечує належну доставку всіх пакетів, перевіряючи їх цілісність і порядок?

4. Який тип IPv6-адреси використовується для надсилання пакету на всі інтерфейси, які належать до групи хостів?

Дослідницькі вправи

1. Наведіть 4 приклади служб, які за умовчанням використовують протокол TCP.

2. Як називається поле в заголовку пакета IPv6, яке реалізує той самий ресурс, що TTL на IPv4?

3. Яку інформацію може виявити Протокол виявлення сусідів (NDP)?

Підсумки

У цьому уроці розглядалися основні транспортні протоколи та служби, що використовуються в стеку TCP/IP.

Ще однією важливою темою була версія 6 IP-протоколу, включаючи адреси IPv6 і основні відмінності від IPv4.

Були розглянуті наступні питання:

- Кореляція між номерами портів і службами.
- TCP (протокол керування передачею).
- UDP (протокол дейтаграм користувача).
- ICMP (міжмережевий протокол керуючих повідомлень).
- Адреса IPv6 і варіанти її скорочення.
- Типи IPv6-адрес.
- Основні відмінності між IPv4 і IPv6.

Відповіді до вправ посібника

1. Який порт є стандартним для протоколу SMTP?

   25

2. Скільки різних портів доступно в системі?

   65535

3. Який транспортний протокол забезпечує належну доставку всіх пакетів, перевіряючи їх цілісність і порядок?

   TCP

4. Який тип IPv6-адреси використовується для надсилання пакету на всі інтерфейси, які належать до групи хостів?

   Груповий

Відповіді до дослідницьких вправ

1. Наведіть 4 приклади служб, які за умовчанням використовують протокол TCP.

   FTP, SMTP, HTTP, POP3, IMAP, SSH

2. Як називається поле в заголовку пакета IPv6, яке реалізує той самий ресурс, що TTL на IPv4?

   Hop Limit

3. Яку інформацію може виявити Протокол виявлення сусідів (NDP)?

   NDP може отримувати різноманітну інформацію з мережі, включаючи інші вузли, адреси, що повторюються, маршрути, DNS-сервери, шлюзи тощо.

109.2 Постійна конфігурація мережі

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 109.2

Ваговий коефіцієнт

4

Ключові галузі знань

- Розуміння основної конфігурації TCP/IP-хоста.
- Налаштування конфігурації мережі Ethernet і Wi-Fi за допомогою NetworkManager.
- Знання systemd-networkd.

Частковий список файлів, термінів та утиліт, що використовуються

- `/etc/hostname`
- `/etc/hosts`
- `/etc/nsswitch.conf`
- `/etc/resolv.conf`
- `nmcli`
- `hostnamectl`
- `ifup`
- `ifdown`

109.2 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	109 Основи мережних технологій
Тема:	109.2 Постійна конфігурація мережі
Урок:	1 з 2

Вступ

У будь-якій мережі TCP/IP кожен вузол повинен налаштувати свій мережний адаптер відповідно до вимог мережі, інакше вони не зможуть спілкуватися один з одним. Тому системний адміністратор повинен надати базову конфігурацію, щоб операційна система могла налаштувати відповідний мережний інтерфейс, а також ідентифікувати себе та основні функції мережі під час кожного завантаження.

Мережні налаштування не залежать від операційних систем, але в останніх є власні методи зберігання та застосування цих налаштувань. Системи Linux покладаються на конфігурації, що зберігаються у звичайних текстових файлах у каталозі `/etc`, для підключення до мережі під час завантаження. Варто знати, як ці файли використовуються, щоб уникнути втрати з'єднання через локальну неправильну конфігурацію.

Мережний інтерфейс

Мережний інтерфейс — це термін, за яким операційна система позначає канал зв'язку, налаштований для роботи з мережним обладнанням, під'єднаним до системи, наприклад пристроєм Ethernet або Wi-Fi. Винятком є інтерфейс *loopback*, який операційна система

використовує, коли їй потрібно встановити з'єднання з собою, але головна мета мережевого інтерфейсу — забезпечити маршрут, через який можна надсилати локальні дані та отримувати віддалені дані. Якщо мережний інтерфейс не налаштовано належним чином, операційна система не зможе спілкуватися з іншими машинами в мережі.

У більшості випадків правильні параметри інтерфейсу визначаються за замовчуванням або налаштовуються під час встановлення операційної системи. Тим не менш, ці параметри часто потрібно перевірити або навіть змінити, коли зв'язок не працює належним чином або коли поведінка інтерфейсу вимагає налаштування.

Є багато команд Linux для переліку мережних інтерфейсів, присутніх у системі, але не всі вони доступні у всіх дистрибутивах. Однак команда `ip` є частиною базового набору мережних інструментів, що входить до складу всіх дистрибутивів Linux, і може використовуватися для переліку мережних інтерфейсів. Повна команда для відображення інтерфейсів – `ip link show`:

```
$ 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: enp3s5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT
group default qlen 1000
        link/ether 00:16:3e:8d:2b:5b brd ff:ff:ff:ff:ff:ff
```

Якщо доступно, також можна використати команду `nmcli device`:

```
$ nmcli device
DEVICE      TYPE      STATE      CONNECTION
enp3s5      ethernet  connected  Gigabit Powerline Adapter
lo          loopback  unmanaged  --
```

Команди, наведені в прикладах, не змінюють жодних налаштувань системи, тому їх може виконувати непривілейований користувач. Обидві команди містять два мережні інтерфейси: `lo` (інтерфейс петлі) і `enp3s5` (інтерфейс Ethernet).

Настільні комп'ютери та ноутбуки під керуванням Linux зазвичай мають два або три попередньо визначених мережні інтерфейси, один для віртуального інтерфейсу зворотного зв'язку, а інші призначені мережному обладнанню, знайденому системою. З іншого боку, сервери та мережні пристрої під керуванням Linux можуть мати десятки мережних інтерфейсів, але до всіх них застосовуються однакові принципи. Абстракція, яку

забезпечує операційна система, дозволяє налаштовувати мережні інтерфейси за допомогою тих самих методів, незалежно від апаратного забезпечення, що лежить в основі.

Однак знання деталей базового апаратного забезпечення інтерфейсу може бути корисним для кращого розуміння того, що відбувається, коли зв'язок не працює належним чином. У системі, де доступно багато мережних інтерфейсів, не може бути очевидним, який із них відповідає, наприклад, Wi-Fi, а який — Ethernet. З цієї причини Linux використовує угоду про найменування інтерфейсу, яка допомагає визначити, який мережний інтерфейс відповідає якому пристрою та порту.

Назви інтерфейсів

Старі дистрибутиви Linux називали мережні інтерфейси Ethernet як `eth0`, `eth1` тощо, пронумеровані відповідно до порядку, в якому ядро ідентифікує пристрої. Бездротові інтерфейси отримали назви `wlan0`, `wlan1` тощо. Ця угода про найменування, однак, не пояснює, який конкретно порт Ethernet відповідає, наприклад, інтерфейсу `eth0`. Залежно від того, як апаратне забезпечення було виявлено, можна було навіть поміняти імена двох мережних інтерфейсів після перезавантаження.

Щоб подолати цю неоднозначність, нові системи Linux використовують попередню угоду про іменування мережних інтерфейсів, створюючи більш тісний зв'язок між іменем інтерфейсу та підключенням базового апаратного забезпечення.

У дистрибутивах Linux, які використовують схему іменування systemd, усі назви інтерфейсів починаються з двозначного префікса, який позначає тип інтерфейсу:

`en`
: Ethernet

`ib`
: InfiniBand

`sl`
: IP-адреса послідовної лінії (slip)

`wl`
: Бездротова локальна мережа (WLAN)

`ww`
: Бездротова глобальна мережа (WWAN)

Від вищого до нижчого пріоритету операційна система використовує такі правила для іменування та нумерації мережних інтерфейсів:

1. Називає інтерфейс за індексом, наданим BIOS або мікропрограмою вбудованих пристроїв, наприклад, `eno1`.
2. Називає інтерфейс за індексом слота PCI Express, як надано BIOS або мікропрограмою, наприклад, `ens1`.
3. Називає інтерфейс за його адресою на відповідній шині, наприклад, `enp3s5`.
4. Називає інтерфейс за MAC-адресою інтерфейсу, наприклад, `enx78e7d1ea46da`.
5. Називає інтерфейс, використовуючи встановлену угоду, наприклад, `eth0`.

Правильно припустити, наприклад, що мережний інтерфейс `enp3s5` був названий так, оскільки він не відповідав першим двом методам іменування, тому замість нього використовувалася його адреса у відповідній шині та слоті. Адреса пристрою `03:05.0`, знайдена у виведенні команди `lspci`, показує асоційований пристрій:

```
$ lspci | fgrep Ethernet
03:05.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-8110SC/8169SC Gigabit
Ethernet (rev 10)
```

Мережні інтерфейси створюються самим ядром Linux, але є багато команд, які можна використовувати для взаємодії з ними. Зазвичай конфігурація встановлюється автоматично, і немає необхідності змінювати налаштування вручну. Тим не менш, за допомогою назви інтерфейсу можна вказати ядру, як продовжувати його налаштування, якщо це необхідно.

Управління інтерфейсом

Протягом багатьох років було розроблено декілька програм для взаємодії з мережними функціями, які надає ядро Linux. Хоча стару команду ifconfig все ще можна використовувати для виконання простих конфігурацій інтерфейсу та запитів, тепер вона застаріла через обмежену підтримку інтерфейсів, відмінних від Ethernet. Команду ifconfig було замінено командою ip, яка здатна керувати багатьма іншими аспектами інтерфейсів TCP/IP, такими як маршрути та тунелі.

Численні можливості команди `ip` можуть бути надмірними для більшості звичайних завдань, тому існують допоміжні команди для полегшення активації та налаштування мережних інтерфейсів. Команди `ifup` і `ifdown` можна використовувати для налаштування мережних інтерфейсів на основі визначень інтерфейсів, зазначених у файлі

`/etc/network/interfaces`. Хоча їх можна викликати вручну, зазвичай ці команди виконуються автоматично під час завантаження системи.

Усі мережні інтерфейси, якими керують `ifup` і `ifdown`, мають бути зазначені у файлі `/etc/network/interfaces`. Формат, який використовується у файлі, є простим: рядки, що починаються зі слова `auto`, використовуються для визначення фізичних інтерфейсів, які мають бути викликані, коли `ifup` виконується з опцією `-a`. Назва інтерфейсу має слідувати за словом `auto` у тому самому рядку. Усі інтерфейси, позначені як `auto`, відкриваються під час завантаження в тому порядку, у якому вони вказані.

WARNING

Методи конфігурації мережі, які використовуються `ifup` і `ifdown`, не стандартизовані в усіх дистрибутивах Linux. CentOS, наприклад, зберігає налаштування інтерфейсу в окремих файлах у каталозі `/etc/sysconfig/network-scripts/`, а формат конфігурації, який використовується в них, дещо відрізняється від формату, який використовується в `/etc/network/interfaces`.

Фактична конфігурація інтерфейсу записується в іншому рядку, починаючи зі слова `iface`, за яким іде назва інтерфейсу, назва сімейства адрес, яку використовує інтерфейс, і назва методу, що використовується для налаштування інтерфейсу. У наступному прикладі показано базовий файл конфігурації для інтерфейсів `lo` (loopback) і `enp3s5`:

```
auto lo
iface lo inet loopback

auto enp3s5
iface enp3s5 inet dhcp
```

Сімейство адрес має бути `inet` для мереж TCP/IP, але є також підтримка мереж IPX (`ipx`) і мереж IPv6 (`inet6`). Інтерфейси петлі використовують метод конфігурації `loopback`. За допомогою методу `dhcp` інтерфейс використовуватиме параметри IP, надані сервером DHCP мережі. Параметри з прикладу конфігурації дозволяють виконувати команду `ifup`, використовуючи назву інтерфейсу `enp3s5` як аргумент:

```
# ifup enp3s5
Internet Systems Consortium DHCP Client 4.4.1
Copyright 2004-2018 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/enp3s5/00:16:3e:8d:2b:5b
```

```
Sending on   LPF/enp3s5/00:16:3e:8d:2b:5b
Sending on   Socket/fallback
DHCPDISCOVER on enp3s5 to 255.255.255.255 port 67 interval 4
DHCPOFFER of 10.90.170.158 from 10.90.170.1
DHCPREQUEST for 10.90.170.158 on enp3s5 to 255.255.255.255 port 67
DHCPACK of 10.90.170.158 from 10.90.170.1
bound to 10.90.170.158 -- renewal in 1616 seconds.
```

У цьому прикладі для інтерфейсу `enp3s5` обрано метод `dhcp`, тому команда `ifup` викликала клієнтську програму DHCP для отримання налаштувань IP від сервера DHCP. Так само команду `ifdown enp3s5` можна використати для вимкнення інтерфейсу.

У мережах без DHCP-сервера замість цього можна використовувати метод `static`, а параметри IP-адреси можна вказати вручну в `/etc/network/interfaces`. Наприклад:

```
iface enp3s5 inet static
    address 192.168.1.2/24
    gateway 192.168.1.1
```

Інтерфейси, які використовують метод `static`, не потребують відповідної директиви `auto`, оскільки вони викликаються кожного разу, коли виявляється мережне обладнання.

Якщо той самий інтерфейс має більше одного запису `iface`, тоді всі налаштовані адреси та параметри будуть застосовані під час виклику цього інтерфейсу. Це корисно для налаштування адрес IPv4 і IPv6 на одному інтерфейсі, а також для налаштування кількох адрес одного типу на одному інтерфейсі.

Локальні та віддалені імена

Працююче налаштування TCP/IP – це лише перший крок до повної зручності використання мережі. На додаток до можливості ідентифікувати вузли в мережі за їхніми IP-номерами, система повинна мати можливість ідентифікувати їх за іменами, легшими для розуміння людиною.

Ім'я, за яким система ідентифікує себе, можна настроїти, і це гарна практика визначати його, навіть якщо машина не призначена для приєднання до мережі. Локальна назва часто збігається з мережевою назвою машини, але це не завжди так. Якщо файл `/etc/hostname` існує, операційна система використовуватиме вміст першого рядка як своє локальне ім'я, яке згодом називатиметься просто *hostname*. Рядки, що починаються з `#` всередині `/etc/hostname`, ігноруються.

Файл `/etc/hostname` можна редагувати безпосередньо, але ім'я хоста машини також можна визначити за допомогою команди `hostnamectl`. Якщо надати підкоманду `set-hostname`, команда `hostnamectl` візьме надану назву як аргумент і запише її в `/etc/hostname`:

```
# hostnamectl set-hostname storage
# cat /etc/hostname
storage
```

Ім'я хоста, визначене в `/etc/hostname`, є *статичним* ім'ям хоста, тобто ім'ям, яке використовується для ініціалізації імені хоста системи під час завантаження. Статичне ім'я хоста може бути рядком довільної форми довжиною до 64 символів. Проте рекомендується, щоб він складався лише з символів нижнього регістру ASCII і без пробілів чи крапок. Він також має обмежуватися форматом, дозволеним для міток доменних імен DNS, навіть якщо це не сувора вимога.

Команда `hostnamectl` може встановити ще два типи імен хостів на додаток до статичного імені хоста:

Правильне ім'я хоста

На відміну від статичного імені хоста, правильне ім'я хоста може включати всі види спеціальних символів. Його можна використовувати для встановлення більш описової назви для машини, наприклад, "LAN Shared Storage":

```
# hostnamectl --pretty set-hostname "LAN Shared Storage"
```

Тимчасове ім'я хоста

Використовується, коли статичне ім'я хоста не встановлено або якщо це ім'я локального хоста за умовчанням. Тимчасове ім'я хоста – це зазвичай ім'я, встановлене разом з іншими автоматичними конфігураціями, але його також можна змінити за допомогою команди `hostnamectl`, наприклад.

```
# hostnamectl --transient set-hostname generic-host
```

Якщо параметри `--pretty` і `--transient` не використовуються, тоді для всіх трьох типів імен хостів буде встановлено вказане ім'я. Щоб установити статичне ім'я хоста, але не правильні та тимчасові імена, натомість слід використовувати опцію `--static`. У всіх випадках лише статичне ім'я хоста зберігається у файлі `/etc/hostname`. Команда `hostnamectl` також може бути використана для відображення різних описових та

ідентифікаційних бітів інформації про запущену систему:

```
$ hostnamectl status
     Static hostname: storage
     Pretty hostname: LAN Shared Storage
  Transient hostname: generic-host
           Icon name: computer-server
             Chassis: server
          Machine ID: d91962a957f749bbaf16da3c9c86e093
             Boot ID: 8c11dcab9c3d4f5aa53f4f4e8fdc6318
    Operating System: Debian GNU/Linux 10 (buster)
              Kernel: Linux 4.19.0-8-amd64
        Architecture: x86-64
```

Це типова дія команди `hostnamectl`, тому підкоманду `status` можна опустити.

Що стосується імен віддалених мережних вузлів, існує два основних способи, які операційна система може застосувати для відповідності імен і IP-номерів: використовувати локальне джерело або використовувати віддалений сервер для перекладу імен в IP-номера і навпаки. Методи можуть доповнювати один одного, а їхній порядок пріоритетів визначається у файлі конфігурації *Name Service Switch*: `/etc/nsswitch.conf`. Цей файл використовується системою та програмами для визначення не лише джерел збігів IP-імен, але й джерел, з яких можна отримати інформацію про службу імен у ряді категорій, які називаються *базами даних*.

База даних *hosts* відстежує відображення між іменами хостів і їх номерами. Рядок усередині `/etc/nsswitch.conf`, що починається з `hosts`, визначає служби, відповідальні за надання асоціацій для нього:

```
hosts: files dns
```

У цьому прикладі запису `files` і `dns` — це імена служб, які визначають, як працюватиме процес пошуку імен хостів. Спочатку система шукатиме збіги в локальних файлах, а потім запитуватиме збіги у служби DNS.

Локальний файл для бази даних хостів – це `/etc/hosts`, простий текстовий файл, який пов’язує IP-адреси з іменами хостів, по одному рядку на IP-адресу, наприклад:

```
127.0.0.1 localhost
```

IP-номер *127.0.0.1* є адресою за замовчуванням для інтерфейсу зворотного зв'язку, отже, він асоціюється з іменем *localhost*.

Також можна прив'язати додаткові псевдоніми до однієї IP-адреси. Псевдоніми можуть забезпечувати альтернативне написання, коротші імена хостів і їх слід додавати в кінці рядка, наприклад:

```
192.168.1.10 foo.mydomain.org foo
```

Правила форматування файлу `/etc/hosts`:

- Поля запису розділені будь-якою кількістю пробілів і/або символів табуляції.
- Текст від символу # до кінця рядка є коментарем і ігнорується.
- Імена хостів можуть містити лише буквено-цифрові символи, знаки мінус і крапки.
- Імена хостів повинні починатися з букви і закінчуватися буквено-цифровим символом.

IPv6-адреси також можна додати до `/etc/hosts`. Наступний запис стосується loopback IPv6-адреси:

```
::1 localhost ip6-localhost ip6-loopback
```

Після специфікації служби `files` специфікація `dns` повідомляє системі запитати службу DNS про потрібну асоціацію імені/IP. Набір процедур, відповідальних за цей метод, називається *резолвер*, а його конфігураційний файл – `/etc/resolv.conf`. У наступному прикладі показано загальний файл `/etc/resolv.conf`, який містить записи для загальнодоступних DNS-серверів Google:

```
nameserver 8.8.4.4
nameserver 8.8.8.8
```

Як показано в прикладі, ключове слово `nameserver` вказує на IP-адресу DNS-сервера. Потрібен лише один сервер імен, але можна надати до трьох серверів імен. Додаткові будуть використані як запасні. Якщо немає записів сервера імен, використовується сервер імен за замовчуванням на локальній машині.

Резолвер можна налаштувати на автоматичне додавання домену до імен перед зверненням до них на сервері імен. Наприклад:

```
nameserver 8.8.4.4
nameserver 8.8.8.8
domain mydomain.org
search mydomain.net mydomain.com
```

Запис `domain` встановлює `mydomain.org` як локальне доменне ім'я, тому для запитів імен у межах цього домену буде дозволено використовувати короткі назви відносно локального домену. Запис `search` має подібну мету, але він приймає список доменів, які слід спробувати, коли надається коротке ім'я. За замовчуванням він містить лише ім'я локального домену.

Вправи до посібника

1. За допомогою яких команд можна вивести список наявних у системі мережних адаптерів?

2. Який тип мережного адаптера з назвою інтерфейсу `wlo1`?

3. Яку роль відіграє файл `/etc/network/interfaces` під час завантаження?

4. Який запис у `/etc/network/interfaces` налаштовує інтерфейс `eno1` для отримання його параметрів IP за допомогою DHCP?

Дослідницькі вправи

1. Як за допомогою команди `hostnamectl` змінити лише *статичне* ім'я хоста локальної машини на `firewall`?

2. Які деталі, крім імен хостів, можна змінити командою `hostnamectl`?

3. Який запис у `/etc/hosts` пов'язує імена `firewall` і `router` з IP-адресою `10.8.0.1`?

4. Як можна змінити файл `/etc/resolv.conf`, щоб надсилати всі запити DNS до `1.1.1.1`?

Підсумки

Цей урок розповідає про те, як вносити постійні зміни в конфігурацію локальної мережі за допомогою стандартних файлів і команд Linux. Linux очікує, що налаштування TCP/IP будуть у певних місцях, і може знадобитися змінити їх, якщо параметри за замовчуванням не підходять. На уроці розглядаються такі теми:

- Як Linux ідентифікує мережні інтерфейси.
- Активація інтерфейсу під час завантаження та базова IP-конфігурація.
- Як операційна система пов'язує імена з хостами.

Розглянуті концепції, команди та процедури:

- Правила іменування інтерфейсу.
- Перелік мережних інтерфейсів за допомогою `ip` і `nmcli`.
- Активація інтерфейсу за допомогою `ifup` і `ifdown`.
- Команда `hostnamectl` і файл `/etc/hostname`.
- Файли `/etc/nsswitch.conf`, `/etc/hosts` і `/etc/resolv.conf`.

Відповіді до вправ посібника

1. За допомогою яких команд можна вивести список наявних у системі мережних адаптерів?

   Команди `ip link show`, `nmcli device` і застаріла `ifconfig`.

2. Який тип мережного адаптера з назвою інтерфейсу `wlo1`?

   Назва починається з `wl`, отже, це адаптер бездротової локальної мережі.

3. Яку роль відіграє файл `/etc/network/interfaces` під час завантаження?

   Він має конфігурації, які використовуються командою `ifup` для активації відповідних інтерфейсів під час завантаження.

4. Який запис у `/etc/network/interfaces` налаштовує інтерфейс `eno1` для отримання його параметрів IP за допомогою DHCP?

   Рядок `iface eno1 inet dhcp`.

Відповіді до дослідницьких вправ

1. Як за допомогою команди `hostnamectl` змінити лише *статичне* ім'я хоста локальної машини на `firewall`?

   З параметром `--static`: `hostnamectl --static set-hostname firewall`.

2. Які деталі, крім імен хостів, можна змінити командою `hostnamectl`?

   `hostnamectl` також може встановити піктограму за замовчуванням для локальної машини, тип її шасі, розташування та середовище розгортання.

3. Який запис у `/etc/hosts` пов'язує імена `firewall` і `router` з IP-адресою `10.8.0.1`?

   Рядок `10.8.0.1 firewall router`.

4. Як можна змінити файл `/etc/resolv.conf`, щоб надсилати всі запити DNS до `1.1.1.1`?

   Використання `nameserver 1.1.1.1` як єдиного запису сервера імен.

109.2 Урок 2

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	109 Основи мережних технологій
Тема:	109.2 Постійна конфігурація мережі
Урок:	2 з 2

Вступ

Linux підтримує практично всі мережні технології, які використовуються для підключення серверів, контейнерів, віртуальних машин, робочих столів і мобільних пристроїв. З'єднання між усіма цими мережними вузлами можуть бути динамічними та неоднорідними, що вимагає відповідного керування операційною системою, яка в них працює.

У минулому для різних дистрибутивів розробляли власні індивідуальні рішення для керування динамічною мережною інфраструктурою. Сьогодні такі інструменти, як *NetworkManager* і *systemd*, надають більш комплексні та інтегровані функції для забезпечення всіх конкретних вимог.

NetworkManager

Більшість дистрибутивів Linux використовують демон служби *NetworkManager* для налаштування та контролю мережних підключень системи. Мета NetworkManager — зробити конфігурацію мережі максимально простою та автоматичною. При використанні DHCP, наприклад, NetworkManager організовує зміни маршруту, отримання IP-адреси та

оновлення локального списку DNS-серверів, якщо це необхідно. Якщо доступні як дротове, так і бездротове з'єднання, NetworkManager надає пріоритет дротовому з'єднанню за замовчуванням. NetworkManager намагатиметься підтримувати принаймні одне з'єднання активним весь час, коли це можливо.

NOTE

Запит із використанням DHCP (*протокол динамічної конфігурації вузла*) зазвичай надсилається через мережний адаптер, щойно зв'язок із мережею встановлено. Сервер DHCP, активний у мережі, відповідає налаштуваннями (IP-адреса, маска мережі, маршрут за замовчуванням тощо), які запитувач повинен використовувати для зв'язку через протокол IP.

За замовчуванням демон NetworkManager контролює мережні інтерфейси, не зазначені у файлі `/etc/network/interfaces`. Це робиться для того, щоб не заважати іншим методам налаштування, які також можуть бути присутніми, таким чином змінюючи лише неконтрольовані інтерфейси.

Служба NetworkManager працює у фоновому режимі з правами root і запускає необхідні дії для підтримки системи в режимі онлайн. Звичайні користувачі можуть створювати та змінювати мережні з'єднання з клієнтськими програмами, які, хоча самі не мають привілеїв адміністратора, здатні спілкуватися з основною службою для виконання запитаних дій.

Клієнтські програми для NetworkManager доступні як для командного рядка, так і для графічного середовища. Для останнього клієнтська програма постачається як елемент середовища робочого столу (під такими іменами, як *nm-tray*, *network-manager-gnome*, *nm-applet* або *plasma-nm*), і зазвичай він доступен через піктограму індикатора в кутку панелі робочого столу або з утиліти налаштування системи.

У командному рядку сам NetworkManager має дві клієнтські програми: `nmcli` і `nmtui`. Обидві програми мають однакові основні функції, але `nmtui` має інтерфейс на основі curses, тоді як `nmcli` є більш комплексною командою, яку також можна використовувати в сценаріях. Команда `nmcli` розділяє всі пов'язані з мережею властивості, якими керує NetworkManager, на категорії під назвою *objects*:

`general`

Загальний стан і операції NetworkManager.

`networking`

Загальний контроль мережі.

`radio`

Перемикачі NetworkManager.

`connection`

Підключення NetworkManager.

`device`

Пристрої, якими керує NetworkManager.

`agent`

Секретний агент NetworkManager або polkit-агент.

`monitor`

Відстежує зміни NetworkManager.

Ім'я об'єкта є основним аргументом команди `nmcli`. Щоб показати загальний статус зв'язку системи, наприклад, об'єкт `general` слід надати як аргумент:

```
$ nmcli general
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN
connected  full          enabled  enabled  enabled  enabled
```

Стовпець `STATE` повідомляє, чи підключена система до мережі чи ні. Якщо з'єднання обмежено через неправильну зовнішню конфігурацію або обмеження доступу, у стовпці `CONNECTIVITY` не повідомлятиметься про статус `full` підключення. Якщо в стовпці `CONNECTIVITY` з'являється `Portal`, це означає, що для завершення процесу підключення потрібні додаткові кроки автентифікації (зазвичай через веб-браузер). Решта стовпців повідомляють про стан бездротових з'єднань (якщо такі є), або `WIFI`, або `WWAN` (Wide Wireless Area Network, тобто стільникові мережі). Суфікс `HW` вказує на те, що стан відповідає мережному пристрою, а не мережному з'єднанню системи, тобто він повідомляє, увімкнено або вимкнено обладнання для економії енергії.

На додаток до аргументу об'єкта, `nmcli` також потребує аргумент команди для виконання. Команда `status` використовується за умовчанням, якщо команда не має аргументу. Тому команда `nmcli general` насправді інтерпретується як `nmcli general status`.

Навряд чи потрібно виконувати будь-які дії, коли мережний адаптер підключений безпосередньо до точки доступу через кабелі, але бездротові мережі потребують подальшої взаємодії, щоб приймати нових учасників. `nmcli` полегшує процес підключення та зберігає налаштування для автоматичного підключення в майбутньому, отже, це дуже

корисно для ноутбуків або інших мобільних пристроїв.

Перш ніж підключатися до Wi-Fi, зручно спочатку вивести перелік доступних мереж в локальній зоні. Якщо в системі є робочий адаптер Wi-Fi, то об'єкт `device` використовуватиме його для сканування доступних мереж за допомогою команди `nmcli device wifi list`:

```
$ nmcli device wifi list
IN-USE  BSSID              SSID        MODE   CHAN  RATE        SIGNAL  BARS  SECURITY
        90:F6:52:C5:FA:12  Hypnotoad   Infra  11    130 Mbit/s  67      ▂▄▆_  WPA2
        10:72:23:C7:27:AC  Jumbao      Infra  1     130 Mbit/s  55      ▂▄__  WPA2
        00:1F:33:33:E9:BE  NETGEAR     Infra  1     54 Mbit/s   35      ▂▄__  WPA1 WPA2
        A4:33:D7:85:6D:B0  AP53        Infra  11    130 Mbit/s  32      ▂▄__  WPA1 WPA2
        98:1E:19:1D:CC:3A  Bruma       Infra  1     195 Mbit/s  22      ▂___  WPA1 WPA2
```

Більшість користувачів, ймовірно, використовуватимуть ім'я в стовпці `SSID`, щоб ідентифікувати мережу, яка їх цікавить. Наприклад, команда `nmcli` може знову підключитися до мережі під назвою `Hypnotoad` за допомогою об'єкта `device`:

```
$ nmcli device wifi connect Hypnotoad
```

Якщо команда виконується в емуляторі терміналу в графічному середовищі, то з'явиться діалогове вікно із запитом на пароль мережі. Під час виконання в текстовій консолі пароль може надаватися разом з іншими аргументами:

```
$ nmcli device wifi connect Hypnotoad password MyPassword
```

Якщо мережа Wi-Fi приховує своє SSID-ім'я, `nmcli` все одно може підключитися до неї за допомогою додаткових аргументів `hidden yes`:

```
$ nmcli device wifi connect Hypnotoad password MyPassword hidden yes
```

Якщо в системі є більше одного адаптера Wi-Fi, той, який буде використовуватися, може бути вказано за допомогою `ifname`. Це приклад підключення за допомогою адаптера з назвою `wlo1`:

```
$ nmcli device wifi connect Hypnotoad password MyPassword ifname wlo1
```

Після успішного з'єднання NetworkManager дасть йому назву за відповідним SSID (якщо це з'єднання Wi-Fi) і збереже його для майбутніх з'єднань. Назви підключень та їхні UUID виводяться за допомогою команди `nmcli connection show`:

```
$ nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Ethernet            53440255-567e-300d-9922-b28f0786f56e  ethernet  enp3s5
tun0                cae685e1-b0c4-405a-8ece-6d424e1fb5f8  tun       tun0
Hypnotoad           6fdec048-bcc5-490a-832b-da83d8cb7915  wifi      wlo1
4G                  a2cf4460-0cb7-42e3-8df3-ccb927f2fd88  gsm       --
```

Показано тип кожного з'єднання, яке може бути `ethernet`, `wifi`, `tun`, `gsm`, `bridge` тощо, а також пристрій, з яким вони пов'язані. Щоб виконувати дії з певним підключенням, потрібно вказати його ім'я або UUID. Щоб вимкнути підключення `Hypnotoad` треба виконати наступне:

```
$ nmcli connection down Hypnotoad
Connection 'Hypnotoad' successfully deactivated
```

Подібним чином можна використати команду `nmcli connection up Hypnotoad`, щоб відкрити з'єднання, оскільки тепер воно збережено NetworkManager. Ім'я інтерфейсу також можна використовувати для повторного підключення, але в цьому випадку замість нього слід використовувати об'єкт `device`:

```
$ nmcli device disconnect wlo2
Device 'wlo1' successfully disconnected.
```

Ім'я інтерфейсу також можна використовувати для відновлення з'єднання:

```
$ nmcli device connect wlo2
Device 'wlo1' successfully activated with '833692de-377e-4f91-a3dc-d9a2b1fcf6cb'.
```

Зауважте, що UUID з'єднання змінюється щоразу, коли відкривається з'єднання, тому для узгодженості бажано використовувати його ім'я.

Якщо бездротовий адаптер доступний, але не використовується, його можна вимкнути для економії енергії. Цього разу об'єкт *radio* слід передати до `nmcli`:

```
$ nmcli radio wifi off
```

Звичайно, бездротовий пристрій можна знову увімкнути командою `nmcli radio wifi on`.

Після встановлення підключень у майбутньому не буде потрібно ручне введення, оскільки NetworkManager визначає доступні відомі мережі та автоматично підключається до них. За потреби NetworkManager має плагіни, які можуть розширити його функціональні можливості, наприклад, плагін для підтримки VPN-з'єднань.

systemd-networkd

Системи, на яких запущено systemd, можуть додатково використовувати вбудовані демони для керування підключенням до мережі: `systemd-networkd` для керування мережевими інтерфейсами та `systemd-resolved` для керування локальним вирішенням імен. Ці служби зворотно сумісні із застарілими методами конфігурації Linux, але конфігурація мережевих інтерфейсів має особливості, про які варто знати.

Конфігураційні файли, які використовує systemd-networkd для налаштування мережних інтерфейсів, можна знайти в будь-якому з наступних трьох каталогів:

`/lib/systemd/network`

Мережний каталог системи.

`/run/systemd/network`

Незалежний мережний каталог середовища виконання.

`/etc/systemd/network`

Каталог локальної мережі адміністрування.

Файли обробляються в лексикографічному порядку, тому їх назви рекомендується починати з цифр, щоб порядок був легшим для читання та налаштування.

Файли в `/etc` мають найвищий пріоритет, тоді як файли в `/run` мають пріоритет над файлами з такою ж назвою в `/lib`. Це означає, що якщо файли конфігурації в різних каталогах мають однакові назви, тоді systemd-networkd ігноруватиме файли з меншим пріоритетом. Подібне розділення файлів — це спосіб змінити параметри інтерфейсу без необхідності змінювати вихідні файли: зміни можна розмістити в `/etc/systemd/network`, щоб замінити зміни в `/lib/systemd/network`.

Призначення кожного файлу конфігурації залежить від його суфікса. Імена файлів, що

закінчуються на `.netdev`, використовуються systemd-networkd для створення віртуальних мережевих пристроїв, таких як пристрої *bridge* або *tun*. Файли із закінченням `.link` встановлюють низькорівневі конфігурації для відповідного мережного інтерфейсу. systemd-networkd виявляє та налаштовує мережні пристрої автоматично, коли вони з'являються, а також ігнорує пристрої, налаштовані іншими засобами, тому в більшості ситуацій немає необхідності додавати ці файли.

Найважливішим суфіксом є `.network`. Файли, які використовують цей суфікс, можна використовувати для налаштування мережних адрес і маршрутів. Як і в інших типах конфігураційних файлів, ім'я файлу визначає порядок, у якому файл буде оброблено. Мережний інтерфейс, на який посилається файл конфігурації, визначається в розділі `[Match]` всередині файлу.

Наприклад, мережний інтерфейс Ethernet `enp3s5` можна вибрати у файлі `/etc/systemd/network/30-lan.network` за допомогою запису `Name=enp3s5` у розділі `[Match]`:

```
[Match]
Name=enp3s5
```

Список імен, розділених пробілами, також приймається для відповідності багатьом мережним інтерфейсам із цим самим файлом одночасно. Імена можуть містити шаблони у стилі оболонки, наприклад, `en*`. Інші записи надають різні правила відповідності, наприклад, вибір мережного пристрою за його MAC-адресою:

```
[Match]
MACAddress=00:16:3e:8d:2b:5b
```

Параметри пристрою знаходяться в розділі `[Network]` файлу. Для простої статичної конфігурації мережі потрібні лише записи `Address` і `Gateway`:

```
[Match]
MACAddress=00:16:3e:8d:2b:5b

[Network]
Address=192.168.0.100/24
Gateway=192.168.0.1
```

Щоб використовувати протокол DHCP замість статичних IP-адрес, слід використовувати

запис `DHCP`:

```
[Match]
MACAddress=00:16:3e:8d:2b:5b

[Network]
DHCP=yes
```

Служба systemd-networkd намагатиметься отримати адреси IPv4 та IPv6 для мережного інтерфейсу. Щоб використовувати лише IPv4, слід записати `DHCP=ipv4`. Подібним чином `DHCP=ipv6` ігноруватиме налаштування IPv4 і використовуватиме лише надану IPv6-адресу.

Захищені паролем бездротові мережі також можуть бути налаштовані за допомогою systemd-networkd, але мережний адаптер має бути вже автентифікований у мережі, перш ніж systemd-networkd зможе його налаштувати. Автентифікацію виконує *WPA supplicant*, програма, призначена для налаштування мережних адаптерів для мереж, захищених паролем.

Першим кроком є створення файлу облікових даних за допомогою команди `wpa_passphrase`:

```
# wpa_passphrase MyWifi > /etc/wpa_supplicant/wpa_supplicant-wlo1.conf
```

Ця команда візьме парольну фразу для бездротової мережі `MyWifi` зі стандартного введення та збереже її хеш у `/etc/wpa_supplicant/wpa_supplicant-wlo1.conf`. Зауважте, що назва файлу має містити відповідну назву бездротового інтерфейсу, отже, `wlo1` у назві файлу.

Менеджер systemd читає файли парольної фрази WPA в `/etc/wpa_supplicant/` і створює відповідну службу для запуску WPA-запитувача та запуску інтерфейсу. Файл парольної фрази, створений у прикладі, матиме відповідну одиницю служби під назвою `wpa_supplicant@wlo1.service`. Команда `systemctl start wpa_supplicant@wlo1.service` пов'яже бездротовий адаптер із віддаленою точкою доступу. Команда `systemctl enable wpa_supplicant@wlo1.service` робить асоціацію автоматичною під час завантаження.

Нарешті, файл `.network`, що відповідає інтерфейсу `wlo1`, має бути присутнім у `/etc/systemd/network/`, оскільки systemd-networkd використовуватиме його для налаштування інтерфейсу, щойно WPA-запитувач завершить асоціацію з точкою доступу.

Вправи до посібника

1. Що означає слово `Portal` у стовпці `CONNECTIVITY` у виведенні команди `nmcli general status`?

2. Як звичайний користувач може використовувати команду `nmcli` у консольному терміналі для підключення до бездротової мережі `MyWifi`, захищеної паролем `MyPassword`?

3. Якою командою можна ввімкнути бездротовий адаптер, якщо він раніше був відключений операційною системою?

4. В якому каталозі слід розміщувати файли спеціальної конфігурації, коли systemd-networkd керує мережними інтерфейсами?

Дослідницькі вправи

1. Як користувач може виконати команду `nmcli`, щоб видалити невикористане підключення під назвою `Hotel Internet`?

2. NetworkManager періодично сканує мережі Wi-Fi, а команда `nmcli device wifi list` виводить лише точки доступу, знайдені під час останнього сканування. Як слід використовувати команду `nmcli`, щоб вказати NetworkManager відразу повторно просканувати всі доступні точки доступу?

3. Який запис `name` слід використовувати в розділі `[Match]` конфігураційного файлу systemd-networkd, щоб відповідати всім інтерфейсам Ethernet?

4. Як слід виконати команду `wpa_passphrase`, щоб використовувати парольну фразу, надану як аргумент, а не зі стандартного введення?

Підсумки

Цей урок охоплює загальні інструменти, які використовуються в Linux для керування різнорідними та динамічними мережними з'єднаннями. Хоча більшість методів конфігурації не вимагають втручання користувача, іноді це необхідно, і такі інструменти, як *NetworkManager* і *systemd-networkd*, можуть звести зусилля до мінімуму. На уроці розглянуто такі теми:

- Як NetworkManager і systemd-networkd інтегруються з системою.
- Як користувач може взаємодіяти з NetworkManager і systemd-networkd.
- Базова конфігурація інтерфейсу з NetworkManager і systemd-networkd.

Розглянуті технології, команди та процедури:

- Команди клієнта NetworkManager: `nmtui` і `nmcli`.
- Сканування та підключення до бездротових мереж за допомогою відповідних команд `nmcli`.
- Постійні підключення до мережі Wi-Fi за допомогою systemd-networkd.

Відповіді до вправ посібника

1. Що означає слово `Portal` у стовпці `CONNECTIVITY` у виведенні команди `nmcli general status`?

   Це означає, що для завершення процесу підключення потрібні додаткові кроки автентифікації (зазвичай через веб-браузер).

2. Як звичайний користувач може використовувати команду `nmcli` у консольному терміналі для підключення до бездротової мережі `MyWifi`, захищеної паролем `MyPassword`?

   У текстовому терміналі команда буде

   ```
   $ nmcli device wifi connect MyWifi password MyPassword
   ```

3. Якою командою можна ввімкнути бездротовий адаптер, якщо він раніше був відключений операційною системою?

   ```
   $ nmcli radio wifi on
   ```

4. В якому каталозі слід розміщувати файли спеціальної конфігурації, коли systemd-networkd керує мережними інтерфейсами?

   У каталозі локальної мережі адміністрування: `/etc/systemd/network`.

Відповіді до дослідницьких вправ

1. Як користувач може виконати команду `nmcli`, щоб видалити невикористане підключення під назвою `Hotel Internet`?

   ```
   $ nmcli connection delete "Hotel Internet"
   ```

2. NetworkManager періодично сканує мережі Wi-Fi, а команда `nmcli device wifi list` виводить лише точки доступу, знайдені під час останнього сканування. Як слід використовувати команду `nmcli`, щоб вказати NetworkManager відразу повторно просканувати всі доступні точки доступу?

   Користувач root може запустити `nmcli device wifi rescan`, щоб змусити NetworkManager повторно сканувати доступні точки доступу.

3. Який запис `name` слід використовувати в розділі `[Match]` конфігураційного файлу systemd-networkd, щоб відповідати всім інтерфейсам Ethernet?

   Запис `name=en*`, оскільки `en` є префіксом для інтерфейсів Ethernet у Linux, а systemd-networkd приймає оболонкові шаблони.

4. Як слід виконати команду `wpa_passphrase`, щоб використовувати парольну фразу, надану як аргумент, а не зі стандартного введення?

   Пароль слід вказувати одразу після SSID, як у `wpa_passphrase MyWifi MyPassword`.

109.3 Основи усунення несправностей мережі

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 109.3

Ваговий коефіцієнт

4

Ключові галузі знань

- Налаштування мережних інтерфейсів вручну, включаючи перегляд і зміну конфігурації мережних інтерфейсів за допомогою iproute2.
- Налаштування маршрутизації вручну, включаючи перегляд і зміну таблиць маршрутизації та встановлення маршруту за замовчуванням за допомогою iproute2.
- Усунення проблем, пов'язаних з конфігурацією мережі.
- Обізнаність із застарілими командами net-tools.

Частковий список файлів, термінів та утиліт, що використовуються

- `ip`
- `hostname`
- `ss`
- `ping`
- `ping6`
- `traceroute`
- `traceroute6`
- `tracepath`
- `tracepath6`

- `netcat`
- `ifconfig`
- `netstat`
- `route`

109.3 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	109 Основи мереж
Тема:	109.3 Основи усунення несправностей мережі
Урок:	1 з 2

Вступ

Linux має дуже гнучкі та потужні мережні можливості. Насправді операційні системи на базі Linux часто використовуються на звичайних мережних пристроях, включаючи дороге комерційне обладнання. Мережі на основі Linux самі по собі можуть мати сертифікацію. З огляду на це, цей урок охопить лише кілька базових інструментів налаштування та усунення несправностей.

Перед цим уроком обов'язково перегляньте уроки про протоколи Інтернету та постійну конфігурацію мережі. У цьому уроці ми розглянемо інструменти для налаштування та усунення несправностей мереж IPv4 і IPv6.

Хоча це не офіційна мета, *пакетні аналізатори*, такі як `tcpdump`, є корисними інструментами для усунення несправностей. Сніфери пакетів дозволяють переглядати та записувати пакети, що надходять до мережного інтерфейсу або виходять з нього. Такі інструменти, як *hex viewers* і *protocol analysers* можна використовувати для перегляду цих пакетів більш детально, ніж це зазвичай дозволяє аналізатор пакетів. Не завадило б хоча б знати про такі програми.

Про команду `ip`

Команда `ip` — це досить нова утиліта, яка використовується для перегляду та налаштування майже будь-чого, що стосується мережних конфігурацій. У цьому уроці розглядаються деякі з підкоманд `ip`, які найбільш часто використовуються, але це тільки верхній шар того, що доступно. Навчившись читати документацію, ви зможете працювати з цією командою набагато ефективніше.

Кожна підкоманда `ip` має власну man-сторінку. У розділі `SEE ALSO` команди `ip` є список цих man-сторінок:

```
$ man ip
...
SEE ALSO
       ip-address(8), ip-addrlabel(8), ip-l2tp(8), ip-link(8), ip-maddress(8),
       ip-monitor(8), ip-mroute(8), ip-neighbour(8), ip-netns(8), ip-
       ntable(8), ip-route(8), ip-rule(8), ip-tcp_metrics(8), ip-token(8), ip-
       tunnel(8), ip-xfrm(8)
       IP Command reference ip-cref.ps
...
```

Замість того, щоб переглядати це кожного разу, коли вам потрібна man-сторінка, просто додайте `-` та назву підкоманди до `ip`, наприклад, `man ip-route`.

Іншим джерелом інформації є функція довідки. Щоб переглянути вбудовану довідку, додайте `help` після підкоманди:

```
$ ip address help
Usage: ip address {add|change|replace} IFADDR dev IFNAME [ LIFETIME ]
                                                      [ CONFFLAG-LIST ]
       ip address del IFADDR dev IFNAME [mngtmpaddr]
       ip address {save|flush} [ dev IFNAME ] [ scope SCOPE-ID ]
                            [ to PREFIX ] [ FLAG-LIST ] [ label LABEL ] [up]
       ip address [ show [ dev IFNAME ] [ scope SCOPE-ID ] [ master DEVICE ]
                         [ type TYPE ] [ to PREFIX ] [ FLAG-LIST ]
                         [ label LABEL ] [up] [ vrf NAME ] ]
       ip address {showdump|restore}
IFADDR := PREFIX | ADDR peer PREFIX
...
```

Огляд маски мережі та маршрутизації

IPv4 і IPv6 відомі як протоколи маршрутизації. Це означає, що вони розроблені таким чином, щоб розробники мереж могли контролювати потік трафіку. Ethernet не є протоколом маршрутизації. Це означає, що якщо ви з'єднаєте купу пристроїв, використовуючи лише Ethernet, ви дуже мало зможете зробити, щоб контролювати потік мережного трафіку.

Протоколи маршрутизації дозволяють розробникам мереж сегментувати мережі, щоб зменшити вимоги до обробки пристроїв підключення, забезпечити резервування та керувати трафіком.

Адреси IPv4 і IPv6 мають дві частини. Перший набір бітів становить мережну частину, а другий набір – вузлову частину. Кількість бітів, які складають частину мережі, визначається *мережною маскою* (також називається *subnet mask*). Іноді її також називають *довжиною префікса*. Незалежно від того, як це називається, це кількість бітів, які машина розглядає як мережну частину адреси. Для IPv4 іноді це вказується в десятковій нотації з крапкою.

Нижче наведено приклад використання IPv4. Зверніть увагу, як двійкові цифри зберігають своє розрядне значення в октетах, навіть якщо їх розділити маскою мережі.

```
192.168.130.5/20

          192      168      130      5
          11000000 10101000 10000010 00000101

20 bits = 11111111 11111111 11110000 00000000

Network = 192.168.128.0
Host    = 2.5
```

Мережна частина адреси використовується машинами IPv4 або IPv6 для пошуку в їх таблицях маршрутизації інтерфейсу, через який потрібно надіслати пакет. Коли хост IPv4 або IPv6 із увімкненою маршрутизацією отримує пакет, який не призначений самому хосту, він намагається зіставити мережну частину пункту призначення з мережею в таблиці маршрутизації. Якщо відповідний запис знайдено, він надсилає пакет до пункту призначення, указаного в таблиці маршрутизації. Якщо записів не знайдено, а маршрут за замовчуванням налаштовано, пакет надсилається за маршрутом за замовчуванням. Якщо запис не знайдено і маршрут за замовчуванням не налаштовано, пакет відхиляється.

Налаштування інтерфейсу

Ми розглянемо два інструменти, які можна використовувати для налаштування мережного інтерфейсу: `ifconfig` та `ip`. Програма ifconfig, хоча й досі широко використовується, вважається застарілим інструментом і може бути недоступною в нових системах.

TIP У нових дистрибутивах Linux встановлення пакунку `net-tools` надасть вам застарілі мережні команди.

Перш ніж налаштовувати інтерфейс, ви повинні спочатку дізнатись, які інтерфейси доступні. Є кілька способів зробити це. Одним із способів є використання опції `-a` для команди `ifconfig`:

```
$ ifconfig -a
```

Інший спосіб – за допомогою `ip`. Іноді ви побачите приклади з `ip addr`, `ip a`, а деякі з `ip address`. Вони є синонімами. Офіційно підкомандою є `ip address`. Це означає, що якщо ви бажаєте переглянути man-сторінку, ви повинні використовувати `man ip-address`, а не `man ip-addr`.

Підкоманда `link` для `ip` виведе посилання на інтерфейси, доступні для налаштування:

```
$ ip link
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: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
group default qlen 1000
    link/ether 08:00:27:54:18:57 brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
group default qlen 1000
    link/ether 08:00:27:ab:11:3e brd ff:ff:ff:ff:ff:ff
```

Припускаючи, що файлову систему `sys` змонтовано, ви також можете вивести вміст `/sys/class/net`:

```
$ ls /sys/class/net
enp0s3  enp0s8  lo
```

Щоб налаштувати інтерфейс за допомогою `ifconfig`, ви повинні увійти в систему як root або скористатися утилітою, такою як `sudo`, щоб запустити команду з правами root. Дотримуйтеся прикладу нижче:

```
# ifconfig enp1s0 192.168.50.50/24
```

Версія `ifconfig` для Linux є гнучкою щодо того, як ви вказуєте маску підмережі:

```
# ifconfig eth2 192.168.50.50 netmask 255.255.255.0
# ifconfig eth2 192.168.50.50 netmask 0xffffff00
# ifconfig enp0s8 add 2001:db8::10/64
```

Зверніть увагу, як у IPv6 було використано ключове слово `add`. Якщо перед IPv6-адресою не поставити `add`, ви отримаєте повідомлення про помилку.

Наступна команда налаштовує інтерфейс за допомогою `ip`:

```
# ip addr add 192.168.5.5/24 dev enp0s8
# ip addr add 2001:db8::10/64 dev enp0s8
```

З `ip` та сама команда використовується як для IPv4, так і для IPv6.

Налаштування параметрів низького рівня

Команда `ip link` використовується для налаштування параметрів інтерфейсу низького рівня або протоколів, таких як VLAN, ARP або MTU, або для вимкнення інтерфейсу.

Загальним завданням для `ip link` є вимикання або вмикання інтерфейсу. Це також можна зробити за допомогою `ifconfig`:

```
# ip link set dev enp0s8 down
# ip link show dev enp0s8
3: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT group
default qlen 1000
    link/ether 08:00:27:ab:11:3e brd ff:ff:ff:ff:ff:ff
# ifconfig enp0s8 up
# ip link show dev enp0s8
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
group default qlen 1000
```

```
    link/ether 08:00:27:ab:11:3e brd ff:ff:ff:ff:ff:ff
```

Іноді може знадобитися налаштувати MTU інтерфейсу. Як і у випадку з увімкненням/вимкненням інтерфейсів, це можна зробити за допомогою `ifconfig` або `ip link`:

```
# ip link set enp0s8 mtu 2000
# ip link show dev enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2000 qdisc pfifo_fast state UP mode DEFAULT
group default qlen 1000
    link/ether 08:00:27:54:53:59 brd ff:ff:ff:ff:ff:ff
# ifconfig enp0s3 mtu 1500
# ip link show dev enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT
group default qlen 1000
    link/ether 08:00:27:54:53:59 brd ff:ff:ff:ff:ff:ff
```

Таблиця маршрутизації

Усі команди `route`, `netstat -r` і `ip route` можна використовувати для перегляду вашої таблиці маршрутизації. Якщо ви хочете змінити свої маршрути, вам потрібно використовувати `route` або `ip route`. Нижче наведено приклади перегляду таблиці маршрутизації:

```
$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         10.0.2.2        0.0.0.0         UG        0 0          0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
192.168.150.0   0.0.0.0         255.255.255.0   U         0 0          0 enp0s8
$ ip route
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
192.168.150.0/24 dev enp0s8 proto kernel scope link src 192.168.150.200
$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.0.2.2        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
192.168.150.0   0.0.0.0         255.255.255.0   U     0      0        0 enp0s8
```

Зверніть увагу, що немає виведення щодо IPv6. Якщо ви бажаєте переглянути свою таблицю маршрутизації для IPv6, ви повинні використовувати `route -6`, `netstat -6r` та `ip -6 route`.

```
$ route -6
Kernel IPv6 routing table
Destination                    Next Hop                   Flag Met Ref Use If
2001:db8::/64                  [::]                       U    256 0      0 enp0s8
fe80::/64                      [::]                       U    100 0      0 enp0s3
2002:a00::/24                  [::]                       !n   1024 0      0 lo
[::]/0                         2001:db8::1                UG   1   0      0 enp0s8
localhost/128                  [::]                       Un   0   2     84 lo
2001:db8::10/128               [::]                       Un   0   1      0 lo
fe80::a00:27ff:fe54:5359/128   [::]                       Un   0   1      0 lo
ff00::/8                       [::]                       U    256 1      3 enp0s3
ff00::/8                       [::]                       U    256 1      6 enp0s8
```

Приклад `netstat -r6` пропущено, оскільки його вихід ідентичний `route -6`. Деякі результати наведеної вище команди `route` є зрозумілими. Стовпець `Flag` містить певну інформацію про маршрут. Прапор `U` вказує на те, що маршрут готовий до використання. `!` означає відхилення маршруту, тобто маршрут із `!` не використовуватиметься. Прапорець `n` означає, що маршрут не кешовано. Ядро підтримує кеш маршрутів для швидшого пошуку окремо від усіх відомих маршрутів. Прапор `G` вказує на шлюз. Стовпець `Metric` або `Met` не використовується ядром. Це стосується адміністративної відстані до цілі. Ця адміністративна відстань використовується протоколами маршрутизації для визначення динамічних маршрутів. Стовпець `Ref` — це кількість посилань або кількість разів використання маршруту. Як і `Metric`, він не використовується ядром Linux. Стовпець `Use` показує кількість пошуків для маршруту.

У вихідних даних `netstat -r` `MSS` вказує на максимальний розмір сегмента для з'єднань TCP за цим маршрутом. Стовпець `Window` показує розмір вікна TCP за умовчанням. `irtt` показує час проходження туди й назад для пакетів на цьому маршруті.

Вихідні дані `ip route` та `ip -6 route` виглядають наступним чином:

1. Пункт призначення.
2. Додатково адреса, за якою зазначена назва інтерфейсу.
3. Протокол маршрутизації, який використовується для додавання маршруту.
4. Охоплення маршруту. Якщо це опущено, це глобальна область або шлюз.
5. Метрика маршруту. Вона використовується протоколами динамічної маршрутизації

для визначення вартості маршруту. Це не використовується більшістю систем.

6. Якщо це маршрут IPv6, перевага маршруту RFC4191.

На кількох прикладах прояснимо це:

Приклад для IPv4

```
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
```

1. Пунктом призначення є маршрут за замовчуванням.
2. Адреса шлюзу `10.0.2.2` доступна через інтерфейс `enp0s3`.
3. Маршрут було додано до таблиці маршрутизації DHCP.
4. Область опущено, тому вона глобальна.
5. Вартість маршруту дорівнює 100.
6. Немає параметрів маршруту IPv6.

приклад для IPv6

```
fc0::/64 dev enp0s8 proto kernel metric 256 pref medium
```

1. Місцем призначення є `fc0::/64`.
2. Він доступний через інтерфейс `enp0s8`.
3. Він був доданий автоматично ядром.
4. Область опущено, тому вона глобальна.
5. Маршрут має вартість `256`.
6. Він має параметр IPv6 `medium`.

Керування маршрутами

Маршрутами можна керувати за допомогою `route` або `ip route`. Нижче наведено приклад додавання та видалення маршруту за допомогою команди `route`. З `route` ви повинні використовувати опцію `-6` для IPv6:

```
# ping6 -c 2 2001:db8:1::20
connect: Network is unreachable
# route -6 add 2001:db8:1::/64 gw 2001:db8::3
# ping6 -c 2 2001:db8:1::20
```

```
PING 2001:db8:1::20(2001:db8:1::20) 56 data bytes
64 bytes from 2001:db8:1::20: icmp_seq=1 ttl=64 time=0.451 ms
64 bytes from 2001:db8:1::20: icmp_seq=2 ttl=64 time=0.438 ms
# route -6 del 2001:db8:1::/64 gw 2001:db8::3
# ping6 -c 2 2001:db8:1::20
connect: Network is unreachable
```

Нижче наведено той самий приклад використання команди `ip route`:

```
# ping6 -c 2 2001:db8:1:20
connect: Network is unreachable
# ip route add 2001:db8:1::/64 via 2001:db8::3
# ping6 -c 2 2001:db8:1:20
PING 2001:db8:1::20(2001:db8:1::20) 56 data bytes
64 bytes from 2001:db8:1::20: icmp_seq=2 ttl=64 time=0.529 ms
64 bytes from 2001:db8:1::20: icmp_seq=2 ttl=64 time=0.438 ms
# ip route del 2001:db8:1::/64 via 2001:db8::3
# ping6 -c 2 2001:db8:1::20
connect: Network is unreachable
```

Вправи до посібника

1. За допомогою яких команд можна вивести список наявних у системі мережних адаптерів?

2. Як би ви тимчасово вимкнули інтерфейс? Як би ви його знову ввімкнули?

3. Що з наведеного нижче є прийнятною маскою підмережі для IPv4?

`0.0.0.255`	
`255.0.255.0`	
`255.252.0.0`	
`/24`	

4. Які команди можна використовувати для перевірки маршруту за замовчуванням?

5. Як додати другу IP-адресу до інтерфейсу?

Дослідницькі вправи

1. Яку підкоманду `ip` можна використати для налаштування тегування vlan?

2. Як би ви налаштували маршрут за замовчуванням?

3. Як отримати детальну інформацію про команду `ip neighbour`? Що станеться, якщо запустити її?

4. Як би ви зробили резервну копію своєї таблиці маршрутизації? Як би ви відновили її?

5. Яку підкоманду `ip` можна використати для налаштування параметрів зв'язуючого дерева?

Підсумки

Мережа зазвичай налаштовується за допомогою сценаріїв запуску системи або помічника, такого як NetworkManager. Більшість дистрибутивів мають інструменти, які редагують файли конфігурації сценарію запуску. Зверніться до документації свого дистрибутива, щоб дізнатися більше.

Можливість ручного налаштування мережі дозволяє ефективніше вирішувати проблеми. Це корисно в мінімальних середовищах, які використовуються для таких речей, як відновлення з резервних копій або перехід на нове обладнання.

Утиліти, розглянуті в цьому розділі, мають більше функціональних можливостей, ніж розглянуті в цьому уроці. Було б доцільно переглянути сторінку довідки кожної з них, щоб ознайомитися з доступними параметрами. Команди `ss` та `ip` — це сучасний спосіб виконання завдань, а решта, які описано, хоча й досі широко використовуються, вважаються застарілими інструментами.

Найкращий спосіб ознайомитися з розглянутими інструментами – це практика. Використовуючи комп'ютер із невеликим об'ємом оперативної пам'яті, можна налаштувати віртуальну мережну лабораторію за допомогою віртуальних машин, з якими можна практикуватися. Трьох віртуальних машин достатньо, щоб зручно працювати з перерахованими інструментами.

Команди, які використані в цьому уроці:

`ifconfig`

Застаріла утиліта, яка використовується для налаштування мережних інтерфейсів і перегляду їх стану.

`ip`

Сучасна і універсальна утиліта, яка використовується для налаштування мережних інтерфейсів і перегляду їх стану.

`netstat`

Застаріла команда, яка використовується для перегляду поточних мережних підключень і інформації про маршрути.

`route`

Застаріла команда, яка використовується для перегляду або зміни таблиці маршрутизації системи.

Відповіді до вправ посібника

1. За допомогою яких команд можна вивести список наявних у системі мережних адаптерів?

   Будь-яка з наведених нижче команд:

   `ip link`, `ifconfig -a` або `ls /sys/class/net`

2. Як би ви тимчасово вимкнули інтерфейс? Як би ви його знову ввімкнули?

   Ви можете використовувати `ifconfig` або `ip link`:

   Використання `ifconfig`:

   ```
   $ ifconfig wlan1 down
   $ ifconfig wlan1 up
   ```

   Використання `ip link`:

   ```
   $ ip link set wlan1 down
   $ ip link set wlan1 up
   ```

3. Що з наведеного нижче є прийнятною маскою підмережі для IPv4?

   - `255.252.0.0`
   - `/24`

     Інші наведені маски недійсні, оскільки вони не розділяють чітко адресу на дві частини, перша частина визначає мережу, а друга – хост. Крайні ліві біти маски завжди будуть `1`, а праві — `0`.

4. Які команди можна використовувати для перевірки маршруту за замовчуванням?

   Ви можете використовувати `route`, `netstat -r` або `ip route`:

   ```
   $ route
   Kernel IP routing table
   Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
   default         server          0.0.0.0         UG    600    0        0 wlan1
   192.168.1.0     0.0.0.0         255.255.255.0   U     600    0        0 wlan1
   ```

```
$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         server          0.0.0.0         UG        0 0          0 wlan1
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 wlan1
$ ip route
default via 192.168.1.20 dev wlan1 proto static metric 600
192.168.1.0/24 dev wlan1 proto kernel scope link src 192.168.1.24 metric 600
```

5. Як додати другу IP-адресу до інтерфейсу?

   Можна використати `ip address` або `ifconfig`. Майте на увазі, що `ifconfig` є застарілим інструментом:

   ```
   $ ip addr add 172.16.15.16/16 dev enp0s9 label enp0s9:sub1
   ```

   Частина команди `label enp0s9:sub1` додає псевдонім до `enp0s9`. Якщо ви не використовуєте застарілий `ifconfig`, ви можете пропустити це. Якщо ви це зробите, команда працюватиме, але адреса, яку ви щойно додали, не відображатиметься під час виведення даних `ifconfig`.

   Ви також можете використовувати `ifconfig`:

   ```
   $ ifconfig enp0s9:sub1 172.16.15.16/16
   ```

Відповіді до дослідницьких вправ

1. Яку підкоманду `ip` можна використати для налаштування тегування vlan?

   `ip link` має параметр `vlan`, який можна використовувати. Нижче наведено приклад позначення підінтерфейсу тегом vlan 20.

   ```
   # ip link add link enp0s9 name enp0s9.20 type vlan id 20
   ```

2. Як би ви налаштували маршрут за замовчуванням?

   Використання `route` або `ip route`:

   ```
   # route add default gw 192.168.1.1
   # ip route add default via 192.168.1.1
   ```

3. Як отримати детальну інформацію про команду `ip neighbour`? Що станеться, якщо запустити її?

   Прочитавши man-сторінку:

   ```
   $ man ip-neigbour
   ```

   Команда відображає ваш ARP-кеш:

   ```
   $ ip neighbour
   10.0.2.2 dev enp0s3 lladdr 52:54:00:12:35:02 REACHABLE
   ```

4. Як би ви зробили резервну копію своєї таблиці маршрутизації? Як би ви відновили її?

   Наведений нижче приклад демонструє резервне копіювання та відновлення таблиці маршрутизації:

   ```
   # ip route save > /root/routes/route_backup
   # ip route restore < /root/routes/route_backup
   ```

5. Яку підкоманду `ip` можна використати для налаштування параметрів зв'язуючого дерева?

Подібно до керування параметрами vlan, `ip link` може налаштувати зв'язуюче дерево за допомогою типу `bridge`. У прикладі показано додавання віртуального інтерфейсу з пріоритетом STP 50:

```
# ip link add link enp0s9 name enp0s9.50 type bridge priority 50
```

109.3 Урок 2

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	109 Основи мереж
Тема:	109.3 Основи усунення несправностей мережі
Урок:	2 з 2

Вступ

Операційні системи на основі Linux мають різноманітні інструменти для вирішення проблем мережі. Цей урок охопить деякі з найпоширеніших. На цьому етапі ви повинні мати уявлення про OSI або інші багаторівневі моделі мереж, адресацію IPv4 або IPv6, а також основи маршрутизації та комутації.

Найкращий спосіб перевірити мережне з'єднання – спробувати використати свій застосунок. Якщо це не спрацьовує, існує багато інструментів, які допоможуть діагностувати проблему.

Тестування з'єднань за допомогою `ping`

Команди `ping` і `ping6` можна використовувати для надсилання exo-запиту ICMP на адресу IPv4 або IPv6 відповідно. Exo-запит ICMP надсилає невелику кількість даних на адресу призначення. Якщо адреса призначення доступна, відправнику буде надіслано повідомлення exo-відповіді ICMP із тими самими даними, які було надіслано в запиті:

```
$ ping -c 3 192.168.50.2
PING 192.168.50.2 (192.168.50.2) 56(84) bytes of data.
64 bytes from 192.168.50.2: icmp_seq=1 ttl=64 time=0.525 ms
64 bytes from 192.168.50.2: icmp_seq=2 ttl=64 time=0.419 ms
64 bytes from 192.168.50.2: icmp_seq=3 ttl=64 time=0.449 ms

--- 192.168.50.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 0.419/0.464/0.525/0.047 ms
```

```
$ ping6 -c 3 2001:db8::10
PING 2001:db8::10(2001:db8::10) 56 data bytes
64 bytes from 2001:db8::10: icmp_seq=1 ttl=64 time=0.425 ms
64 bytes from 2001:db8::10: icmp_seq=2 ttl=64 time=0.480 ms
64 bytes from 2001:db8::10: icmp_seq=3 ttl=64 time=0.725 ms

--- 2001:db8::10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.425/0.543/0.725/0.131 ms
```

Параметр `-c` використовується для визначення кількості пакетів для надсилання. Якщо ви пропустите цей параметр, `ping` і `ping6` продовжуватимуть надсилати пакети, доки ви їх не зупините, як правило, за допомогою комбінації клавіш `Ctrl` + `C`.

Те, що ви не можете пропінгувати хост, не означає, що ви не можете підключитися до нього. У багатьох організаціях є міжмережні екрани або списки контролю доступу на маршрутизаторах, які блокують усе, окрім мінімуму, необхідного для функціонування їхніх систем. В тому числі можуть блокуватись echo-запити та відповіді ICMP. Оскільки ці пакети можуть містити довільні дані, спритний зловмисник може використати їх для викрадання даних.

Трасування маршрутів

Програми `traceroute` і `traceroute6` можна використовувати, щоб показати маршрут, яким пакет дістається до місця призначення. Вони роблять це, надсилаючи кілька пакетів до місця призначення, збільшуючи поле *Time-To-Live* (TTL) заголовка IP з кожним наступним пакетом. Кожен маршрутизатор на цьому шляху відповість повідомленням ICMP про перевищення TTL:

```
$ traceroute 192.168.1.20
```

```
traceroute to 192.168.1.20 (192.168.1.20), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  0.396 ms  0.171 ms  0.132 ms
 2  192.168.1.20 (192.168.1.20)  2.665 ms  2.573 ms  2.573 ms
$ traceroute 192.168.50.2
traceroute to 192.168.50.2 (192.168.50.2), 30 hops max, 60 byte packets
 1  192.168.50.2 (192.168.50.2)  0.433 ms  0.273 ms  0.171 ms
$ traceroute6 2001:db8::11
traceroute to 2001:db8::11 (2001:db8::11), 30 hops max, 80 byte packets
 1  2001:db8::11 (2001:db8::11)  0.716 ms  0.550 ms  0.641 ms
$ traceroute 2001:db8::11
traceroute to 2001:db8::11 (2001:db8::11), 30 hops max, 80 byte packets
 1  2001:db8::10 (2001:db8::11)  0.617 ms  0.461 ms  0.387 ms
$ traceroute net2.example.net
traceroute to net2.example.net (192.168.50.2), 30 hops max, 60 byte packets
 1  net2.example.net (192.168.50.2)  0.533 ms  0.529 ms  0.504 ms
$ traceroute6 net2.example.net
traceroute to net2.example.net (2001:db8::11), 30 hops max, 80 byte packets
 1  net2.example.net (2001:db8::11)  0.738 ms  0.607 ms  0.304 ms
```

За замовчуванням `traceroute` надсилає 3 пакети UDP із небажаними даними на порт 33434, збільшуючи їх щоразу, коли надсилає пакет. Кожен рядок виводу команди є інтерфейсом маршрутизатора, через який проходить пакет. Час, показаний у кожному рядку вихідних даних, є часом проходження для кожного пакету. IP-адреса – це адреса відповідного інтерфейсу маршрутизатора. Якщо `traceroute` може, він використовує DNS-ім'я інтерфейсу маршрутизатора. Іноді ви побачите `*` замість часу. Коли це відбувається, це означає, що `traceroute` ніколи не отримував повідомлення про перевищення TTL для цього пакета. Це часто вказує на те, що остання відповідь є останнім переходом на маршруті.

Якщо у вас є доступ до `root`, параметр `-I` налаштує `traceroute` на використання ехо-запитів ICMP замість UDP-пакунків. Це часто ефективніше, ніж UDP, тому що вузол призначення швидше відповість на ехо-запит ICMP, ніж на пакет UDP:

```
# traceroute -I learning.lpi.org
traceroute to learning.lpi.org (208.94.166.201), 30 hops max, 60 byte packets
 1  047-132-144-001.res.spectrum.com (47.132.144.1)  9.764 ms  9.702 ms  9.693 ms
 2  096-034-094-106.biz.spectrum.com (96.34.94.106)  8.389 ms  8.481 ms  8.480 ms
 3  dtr01hlrgnc-gbe-4-15.hlrg.nc.charter.com (96.34.64.172)  8.763 ms  8.775 ms  8.770 ms
 4  acr01mgtnnc-vln-492.mgtn.nc.charter.com (96.34.67.202)  27.080 ms  27.154 ms  27.151 ms
 5  bbr01gnvlsc-bue-3.gnvl.sc.charter.com (96.34.2.112)  31.339 ms  31.398 ms  31.395 ms
 6  bbr01aldlmi-tge-0-0-0-13.aldl.mi.charter.com (96.34.0.161)  39.092 ms  38.794 ms  38.821
ms
```

```
 7  prr01ashbva-bue-3.ashb.va.charter.com (96.34.3.51)  34.208 ms  36.474 ms  36.544 ms
 8  bx2-ashburn.bell.ca (206.126.236.203)  53.973 ms  35.975 ms  38.250 ms
 9  tcore4-ashburnbk_0-12-0-0.net.bell.ca (64.230.125.190)  66.315 ms  65.319 ms  65.345 ms
10  tcore4-toronto47_2-8-0-3.net.bell.ca (64.230.51.22)  67.427 ms  67.502 ms  67.498 ms
11  agg1-toronto47_xe-7-0-0_core.net.bell.ca (64.230.161.114)  61.270 ms  61.299 ms  61.291
ms
12  dis4-clarkson16_5-0.net.bell.ca (64.230.131.98)  61.101 ms  61.177 ms  61.168 ms
13  207.35.12.142 (207.35.12.142)  70.009 ms  70.069 ms  59.893 ms
14  unassigned-117.001.centrilogic.com (66.135.117.1)  61.778 ms  61.950 ms  63.041 ms
15  unassigned-116.122.akn.ca (66.135.116.122)  62.702 ms  62.759 ms  62.755 ms
16  208.94.166.201 (208.94.166.201)  62.936 ms  62.932 ms  62.921 ms
```

Деякі організації блокують echo-запити та відповіді ICMP. Щоб обійти це, ви можете використовувати TCP. Використовуючи відомий відкритий порт TCP, ви можете гарантувати відповідь хоста призначення. Щоб використовувати TCP, використовуйте опцію `-T` разом із `-p`, щоб зазначити порт. Як і у випадку з echo-запитами ICMP, для цього ви повинні мати доступ як `root`:

```
# traceroute -m 60 -T -p 80  learning.lpi.org
traceroute to learning.lpi.org (208.94.166.201), 60 hops max, 60 byte packets
 1  * * *
 2  096-034-094-106.biz.spectrum.com (96.34.94.106)  12.178 ms  12.229 ms  12.175 ms
 3  dtr01hlrgnc-gbe-4-15.hlrg.nc.charter.com (96.34.64.172)  12.134 ms  12.093 ms  12.062 ms
 4  acr01mgtnnc-vln-492.mgtn.nc.charter.com (96.34.67.202)  31.146 ms  31.192 ms  31.828 ms
 5  bbr01gnvlsc-bue-3.gnvl.sc.charter.com (96.34.2.112)  39.057 ms  46.706 ms  39.745 ms
 6  bbr01aldlmi-tge-0-0-0-13.aldl.mi.charter.com (96.34.0.161)  50.590 ms  58.852 ms  58.841
ms
 7  prr01ashbva-bue-3.ashb.va.charter.com (96.34.3.51)  34.556 ms  37.892 ms  38.274 ms
 8  bx2-ashburn.bell.ca (206.126.236.203)  38.249 ms  36.991 ms  36.270 ms
 9  tcore4-ashburnbk_0-12-0-0.net.bell.ca (64.230.125.190)  66.779 ms  63.218 ms tcore3-
ashburnbk_100ge0-12-0-0.net.bell.ca (64.230.125.188)  60.441 ms
10  tcore4-toronto47_2-8-0-3.net.bell.ca (64.230.51.22)  63.932 ms  63.733 ms  68.847 ms
11  agg2-toronto47_xe-7-0-0_core.net.bell.ca (64.230.161.118)  60.144 ms  60.443 ms agg1-
toronto47_xe-7-0-0_core.net.bell.ca (64.230.161.114)  60.851 ms
12  dis4-clarkson16_5-0.net.bell.ca (64.230.131.98)  67.246 ms dis4-clarkson16_7-
0.net.bell.ca (64.230.131.102)  68.404 ms dis4-clarkson16_5-0.net.bell.ca (64.230.131.98)
67.403 ms
13  207.35.12.142 (207.35.12.142)  66.138 ms  60.608 ms  64.656 ms
14  unassigned-117.001.centrilogic.com (66.135.117.1)  70.690 ms  62.190 ms  61.787 ms
15  unassigned-116.122.akn.ca (66.135.116.122)  62.692 ms  69.470 ms  68.815 ms
16  208.94.166.201 (208.94.166.201)  61.433 ms  65.421 ms  65.247 ms
17  208.94.166.201 (208.94.166.201)  64.023 ms  62.181 ms  61.899 ms
```

Як і `ping`, `traceroute` має свої обмеження. Міжмережні екрани та маршрутизатори можуть блокувати пакети, надіслані з або повернуті до `traceroute`. Якщо у вас є `root` доступ, існують параметри, які можуть допомогти вам отримати точні результати.

Пошук MTU за допомогою `tracepath`

Команда `tracepath` схожа на `traceroute`. Різниця полягає в тому, що він відстежує розміри *максимальної одиниці передачі* (MTU) уздовж шляху. MTU – це налаштований параметр мережного інтерфейсу або апаратне обмеження найбільшого блоку даних протоколу, який він може передавати або отримувати. Програма `tracepath` працює так само, як `traceroute`, оскільки вона збільшує TTL з кожним пакетом. Вона відрізняється надсиланням дуже великої датаграми UDP. Це майже неминуче, щоб датаграма була більшою, ніж MTU для пристрою із найменшим MTU на маршруті. Коли пакет досягає цього пристрою, пристрій, як правило, відповідає пакетом «Адресат недосяжний». Такий ICMP-пакет має поле для MTU з'єднання, в якому пристрій зазначає максимальний розмір пакету, який зміг би переслати. Потім `tracepath` надсилає всі наступні пакети такого розміру:

```
$ tracepath 192.168.1.20
 1?: [LOCALHOST]                                         pmtu 1500
 1:  10.0.2.2                                              0.321ms
 1:  10.0.2.2                                              0.110ms
 2:  192.168.1.20                                          2.714ms reached
     Resume: pmtu 1500 hops 2 back 64
```

На відміну від `traceroute`, ви повинні явно використовувати `tracepath6` для IPv6:

```
$ tracepath 2001:db8::11
tracepath: 2001:db8::11: Address family for hostname not supported
$ tracepath6 2001:db8::11
 1?: [LOCALHOST]                      0.027ms pmtu 1500
 1:  net2.example.net                                      0.917ms reached
 1:  net2.example.net                                      0.527ms reached
     Resume: pmtu 1500 hops 1 back 1
```

Результат подібний до `traceroute`. Перевагою `tracepath` є те, що в останньому рядку він виводить найменший MTU для всього шляху. Це може бути корисним для усунення несправностей підключень, які не можуть обробити фрагменти.

Як і в попередніх інструментах усунення несправностей, обладнання може заблокувати

ваші пакети.

Створення довільних зв'язків

Програма `nc`, відома як netcat, може надсилати або отримувати довільні дані через мережне підключення TCP або UDP. Наступні приклади повинні прояснити її функціональність.

Ось приклад налаштування слухача на порт `1234`:

```
$ nc -l 1234
LPI Example
```

Результати `LPI Example` з'являються після наведеного нижче прикладу, де налаштовується відправник netcat для надсилання пакетів на `net2.example.net` через порт `1234`. Параметр `-l` використовується, щоб вказати, що ви бажаєте, щоб `nc` отримував дані, а не надсилав їх:

```
$ nc net2.example.net 1234
LPI Example
```

Натисніть `Ctrl` + `C` на будь-якій системі, щоб припинити з'єднання.

Netcat працює як з IPv4, так і з IPv6 адресами. Він працює як з TCP, так і з UDP. Його навіть можна використовувати для налаштування грубої віддаленої оболонки.

WARNING

Зауважте, що не кожен інсталяційний пакунок `nc` підтримує перемикач `-e`. Обов'язково перегляньте man-сторінки для вашого інсталяційного пакунку, щоб отримати інформацію про безпеку щодо цього параметра, а також про альтернативні методи виконання команд у віддаленій системі.

```
$ hostname
net2
$ nc -u -e /bin/bash -l 1234
```

Параметр `-u` призначений для UDP. `-e` вказує netcat надсилати все, що він отримує, на стандартне введення виконуваного файлу, яке слідує за ним. У цьому прикладі `/bin/bash`.

```
$ hostname
net1
$ nc -u net2.example.net 1234
hostname
net2
pwd
/home/emma
```

Зверніть увагу, як вихідні дані команди `hostname` збігаються з результатом хоста, що прослуховується, і каталогом, виведеним командою pwd.

Перегляд поточних підключень і слухачів

Програми `netstat` і `ss` можна використовувати для перегляду стану ваших поточних слухачів і з'єднань. Як і `ifconfig`, `netstat` є застарілим інструментом. Як `netstat`, так і `ss` мають подібні результати та параметри. Ось деякі параметри, доступні для обох програм:

`-a`
: Показує всі сокети.

`-l`
: Показує слухаючі сокети.

`-p`
: Показує процес, пов'язаний із з'єднанням.

`-n`
: Запобігає пошуку імен як для портів, так і для адрес.

`-t`
: Показує TCP-з'єднання.

`-u`
: Показує UDP-з'єднання.

У наведених нижче прикладах показано результат набору параметрів, який зазвичай використовується для обох програм:

```
# netstat -tulnp
Active Internet connections (only servers)
```

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program
name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      892/sshd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1141/master
tcp6       0      0 :::22                   :::*                    LISTEN      892/sshd
tcp6       0      0 ::1:25                  :::*                    LISTEN      1141/master
udp        0      0 0.0.0.0:68              0.0.0.0:*                           692/dhclient
# ss -tulnp
# ss -tulnp
Netid  State      Recv-Q Send-Q     Local Address:Port                    Peer
Address:Port
udp    UNCONN     0      0                     :68                                  *:
users:(("dhclient",pid=693,fd=6))
tcp    LISTEN     0      128                   :22                                  *:
users:(("sshd",pid=892,fd=3))
tcp    LISTEN     0      100           127.0.0.1:25                                  :
users:(("master",pid=1099,fd=13))
tcp    LISTEN     0      128               [::]:22                               [::]:*
users:(("sshd",pid=892,fd=4))
tcp    LISTEN     0      100               [::1]:25                              [::]:*
users:(("master",pid=1099,fd=14))
```

Стовпець `Recv-Q` містить кількість пакетів, отриманих сокетом, але не переданих його програмі. Стовпець `Send-Q` містить кількість пакетів, надісланих сокетом, які не були підтверджені одержувачем. Решта стовпців зрозумілі самі по собі.

Вправи до посібника

1. Які команди ви б використали, щоб надіслати ICMP-запити на `learning.lpi.org`?

2. Як ви можете визначити маршрут до `8.8.8.8`?

3. Яка команда покаже вам, якщо якісь процеси прослуховують TCP-порт 80?

4. Як можна дізнатися, який процес прослуховує порт?

5. Як можна визначити максимальний MTU мережевого шляху?

Дослідницькі вправи

1. Як ви можете використовувати netcat для надсилання HTTP-запиту на веб-сервер?

2. З яких причин може не вдатися перевірка доступності хоста за допомогою ping?

3. Назвіть інструмент, який можна використовувати, щоб побачити мережеві пакети, що досягають або залишають хост Linux?

4. Як ви можете змусити `traceroute` використовувати інший інтерфейс?

5. Чи можливо для `traceroute` повідомляти MTU?

Підсумки

Мережа зазвичай налаштовується за допомогою сценаріїв запуску системи або помічника, такого як NetworkManager. Більшість дистрибутивів мають інструменти, які редагують файли конфігурації сценарію запуску. Зверніться до документації свого дистрибутива, щоб дізнатися більше.

Можливість ручного налаштування мережі дозволяє ефективніше вирішувати проблеми. Це корисно в мінімальних середовищах, які використовуються для таких речей, як відновлення з резервних копій або перехід на нове обладнання.

Утиліти, розглянуті в цьому розділі, мають більше функціональних можливостей, ніж розглянуті в цьому уроці. Було б доцільно переглянути сторінку довідки кожної з них, щоб ознайомитися з доступними параметрами. Команди `ss` та `ip` — це сучасний спосіб виконання завдань, а решта, які описано, хоча й досі широко використовуються, вважаються застарілими інструментами.

Найкращий спосіб ознайомитися з розглянутими інструментами – це практика. Використовуючи комп'ютер із невеликим об'ємом оперативної пам'яті, можна налаштувати віртуальну мережну лабораторію за допомогою віртуальних машин, з якими можна практикуватися. Трьох віртуальних машин достатньо, щоб зручно працювати з перерахованими інструментами.

Команди, розглянуті в цьому уроці:

`ping` та `ping6`

Використовуються для передачі ICMP-пакетів на віддалений хост для перевірки доступності мережного з'єднання.

`traceroute` та `traceroute6`

Використовуються для відстеження шляху через мережу для визначення підключення до мережі.

`tracepath` та `tracepath6`

Використовуються для відстеження шляху через мережу, а також визначення розмірів MTU уздовж маршруту.

`nc`

Використовується для встановлення довільних з'єднань у мережі з метою перевірки з'єднання, а також запиту мережі щодо доступних служб і пристроїв.

`netstat`

Застаріла команда, яка використовується для визначення відкритих мережевих підключень системи та статистики.

`ss`

Сучасна команда, яка використовується для визначення відкритих мережевих підключень системи та статистики.

Відповіді до вправ посібника

1. Які команди ви б використали, щоб надіслати ICMP-запити на `learning.lpi.org`?

   Ви б використали `ping` або `ping6`:

   ```
   $ ping learning.lpi.org
   ```

   або

   ```
   $ ping6 learning.lpi.org
   ```

2. Як ви можете визначити маршрут до `8.8.8.8`?

   За допомогою команд `tracepath` або `traceroute`.

   ```
   $ tracepath 8.8.8.8
   ```

   або

   ```
   $ traceroute 8.8.8.8
   ```

3. Яка команда покаже вам, якщо якісь процеси прослуховують TCP-порт 80?

   За допомогою команди `ss`:

   ```
   $ ss -ln | grep ":80"
   ```

   За допомогою команди `netstat`:

   ```
   $ netstat -ln | grep ":80"
   ```

   Хоча це не є обов'язковою умовою для іспиту, ви також можете використовувати lsof:

   ```
   # lsof -Pi:80
   ```

4. Як можна дізнатися, який процес прослуховує порт?

Знову ж таки, є кілька способів зробити це. Ви можете використовувати `lsof` так само, як у попередній відповіді, замінюючи номер порту. Ви також можете використовувати `netstat` або `ss` з опцією `-p`. Пам'ятайте, що `netstat` вважається застарілим інструментом.

```
# netstat -lnp | grep ":22"
```

Ті самі параметри, які працюють з `netstat`, також працюють з `ss`:

```
# ss -lnp | grep ":22"
```

5. Як можна визначити максимальний MTU мережевого шляху?

   За допомогою команди `tracepath`:

```
$ tracepath somehost.example.com
```

Відповіді до дослідницьких вправ

1. Як ви можете використовувати netcat для надсилання HTTP-запиту на веб-сервер?

   Ввівши рядок запиту HTTP, будь-які заголовки та порожній рядок у термінал:

```
$ nc learning.lpi.org 80
GET /index.html HTTP/1.1
HOST: learning.lpi.org

HTTP/1.1 302 Found
Location: https://learning.lpi.org:443/index.html
Date: Wed, 27 May 2020 22:54:46 GMT
Content-Length: 5
Content-Type: text/plain; charset=utf-8

Found
```

2. З яких причин може не вдатися перевірка доступності хоста за допомогою ping?

   Існує кілька можливих причин. Ось деякі з них:

   - Віддалений хост не працює.
   - ACL маршрутизатора блокує ваш ping.
   - Міжмережний екран віддаленого хоста блокує ваш ping.
   - Можливо, ви використовуєте неправильне ім'я або адресу хоста.
   - Ваше розпізнавання імені повертає неправильну адресу.
   - Конфігурація мережі вашого пристрою неправильна.
   - Брандмауер вашого комп'ютера блокує ping.
   - Конфігурація мережі віддаленого хоста неправильна.
   - Інтерфейс(и) вашої машини від'єднано.
   - Інтерфейс(и) віддаленого комп'ютера від'єднано.
   - Компонент мережі, такий як комутатор, кабель або маршрутизатор між вашою машиною і віддаленою, більше не працює.

3. Назвіть інструмент, який можна використовувати, щоб побачити мережеві пакети, що досягають або залишають хост Linux?

Можна використовувати як `tcpdump`, так і `wireshark`.

4. Як ви можете змусити `traceroute` використовувати інший інтерфейс?

   За допомогою параметра `-i`:

```
$ traceroute -i eth2 learning.lpi.org
traceroute -i eth2 learning.lpi.org
traceroute to learning.lpi.org (208.94.166.201), 30 hops max, 60 byte packets
...
```

5. Чи можливо для `traceroute` повідомляти MTU?

   Так, з опцією `--mtu`:

```
# traceroute -I --mtu  learning.lpi.org
traceroute to learning.lpi.org (208.94.166.201), 30 hops max, 65000 byte packets
 1  047-132-144-001.res.spectrum.com (47.132.144.1)  9.974 ms F=1500  10.476 ms  4.743 ms
 2  096-034-094-106.biz.spectrum.com (96.34.94.106)  8.697 ms  9.963 ms  10.321 ms
...
```

109.4 Налаштування DNS на стороні клієнта

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 109.4

Ваговий коефіцієнт

2

Ключові галузі знань

- Створення запитів до віддалених DNS-серверів.
- Налаштування локального дозволу імен і використання віддалених DNS-серверів.
- Зміна порядку розпізнавання імен.
- Помилки налагодження, пов'язані з розпізнаванням імен.
- Обізнаність про systemd-resolved.

Частковий список файлів, термінів та утиліт, що використовуються

- `/etc/hosts`
- `/etc/resolv.conf`
- `/etc/nsswitch.conf`
- `host`
- `dig`
- `getent`

Linux Professional Institute

109.4 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	109 Основи мереж
Тема:	109.4 Налаштування DNS на стороні клієнта
Урок:	1 з 1

Вступ

Цей урок охоплює налаштування розпізнавання імен на стороні клієнта та використання кількох інструментів командного рядка для розпізнавання імен.

Запам’ятати та підтримувати IP-адреси, UID, GID та інші номери для всього неможливо. Служби розпізнавання імен перетворюють імена, які легко запам’ятати, на числа та навпаки. У цьому уроці розглянуто розпізнавання імен хостів, але подібний процес відбувається для імен користувачів, імен груп, номерів портів та деяких інших.

Процес розпізнавання імен

Програми, які перетворюють імена в числа, майже завжди використовують функції стандартної бібліотеки C, яка в системах Linux представлена GNU-проєктом glibc. Перше, що роблять ці функції, це читають файл `/etc/nsswitch.conf`, щоб отримати інструкції щодо того, як розпізнати цей тип імені. Цей урок зосереджений на розпізнаванні імен хостів, але той самий процес стосується й інших типів розпізнавання імен. Коли процес читає `/etc/nsswitch.conf`, він шукає ім’я у вказаний спосіб. Оскільки `/etc/nsswitch.conf` підтримує плагіни, далі може бути що завгодно. Після того, як

функція завершить пошук імені чи номера, вона повертає результат до процесу виклику.

DNS класи

DNS має три класи записів: IN, HS і CH. У цьому уроці всі DNS-запити будуть типу IN. Клас IN призначений для інтернет-адрес, що використовують стек TCP/IP. CH призначений для ChaosNet, яка є мережною технологією, що існувала недовго і більше не використовується. Клас HS для Hesiod. Hesiod — це спосіб зберігати такі речі, як passwd і записи груп у DNS. Hesiod виходить за рамки цього уроку.

Розуміння `/etc/nsswitch.conf`

Найкращий спосіб дізнатися про цей файл — прочитати man-сторінку, яка є частиною проекту man-сторінок Linux. Ця сторінка доступна у більшості систем. Доступ до неї можна отримати за допомогою команди `man nsswitch.conf`. Крім того, її можна знайти за адресою https://man7.org/linux/man-pages/dir_section_5.html

Нижче наведено простий приклад `/etc/nsswitch.conf` з man-сторінки:

```
passwd:         compat
group:          compat
shadow:         compat

hosts:          dns [!UNAVAIL=return] files
networks:       nis [NOTFOUND=return] files
ethers:         nis [NOTFOUND=return] files
protocols:      nis [NOTFOUND=return] files
rpc:            nis [NOTFOUND=return] files
services:       nis [NOTFOUND=return] files
# This is a comment. It is ignored by the resolution functions.
```

Файл організований у стовпці. Крайній лівий стовпець – це тип бази даних імен. Решта стовпців — це методи, які функції розв'язання мають використовувати для пошуку імені. За методами йдуть функції зліва направо. Стовпці з `[]` використовуються для надання деякої обмеженої умовної логіки стовпцю безпосередньо ліворуч від нього.

Припустимо, що процес намагається розпізнати ім'я хоста `learning.lpi.org`. Це зробить відповідний виклик бібліотеки C (швидше за все, `gethostbyname`). Потім ця функція читатиме `/etc/nsswitch.conf`. Оскільки процес шукає ім'я хоста, він знайде рядок, що починається з `hosts`. Потім він спробує використати DNS для визначення імені. Наступний стовпець `[!UNAVAIL=return]` означає, що якщо служба *доступна*, не намагайтеся

використовувати наступне джерело, тобто якщо DNS доступний, припиніть спроби розпізнати ім'я хоста, навіть якщо сервери імен не змогли цього зробити. Якщо DNS недоступний, перейдіть до наступного джерела. У цьому випадку наступним джерелом є `files`.

Коли ви бачите стовпець у форматі `[result=action]`, це означає, що коли пошуком резольвера стовпця ліворуч від нього є `result`, тоді виконується `action`. Якщо перед `result` стоїть `!`, це означає, що якщо результат не є `result`, виконайте `action`. Описи можливих результатів і дій дивись на man-сторінці.

Тепер припустимо, що процес намагається знайти номер порту відповідно до імені служби. Він читатиме рядок `services`. Першим джерелом у списку є NIS. NIS розшифровується як *Network Information Service* (її іноді називають жовтими сторінками). Це стара служба, яка дозволяла централізовано керувати такими речами, як користувачі. Її вже рідко використовують через слабкий захист. Наступний стовпець `[NOTFOUND=return]` означає, що якщо пошук вдався, але службу не знайдено, потрібно припинити пошук. Якщо вищезазначена умова не виконується, використовуйте локальні файли.

Все, що праворуч від #, є коментарем і ігнорується.

Файл `/etc/resolv.conf`

Файл `/etc/resolv.conf` використовується для налаштування пошукової здатності хоста через DNS. Деякі дистрибутиви мають сценарії запуску, демони та інші інструменти, які записують у цей файл. Майте це на увазі під час редагування цього файлу вручну. Перевірте свій дистрибутив і будь-яку документацію інструментів налаштування мережі у такому випадку. Деякі інструменти, наприклад, Network Manager, залишать коментар у файлі, повідомляючи, що внесені вручну зміни буде перезаписано.

Як і у випадку з `/etc/nsswitch.conf`, існує man-сторінка, пов'язана з файлом. Доступ до неї можна отримати за допомогою команди `man resolv.conf` або за адресою https://man7.org/linux/man-pages/man5/resolv.conf.5.html.

Формат файлу досить простий. У крайньому лівому стовпчику ви маєте опцію `name`. Решта стовпців у цьому ж рядку є значенням опції.

Найпоширенішим варіантом є параметр `nameserver`. Він використовується для визначення адреси IPv4 або IPv6 DNS-сервера. На дату написання цієї статті ви можете вказати до трьох серверів імен. Якщо ваш `/etc/resolv.conf` не має опції `nameserver`, ваша система за умовчанням використовуватиме сервер імен на локальній машині.

Нижче наведено простий файл прикладу типових конфігурацій:

```
search lpi.org
nameserver 10.0.0.53
nameserver fd00:ffff::2:53
```

Опція `search` використовується, щоб дозволити пошук у короткій формі. У прикладі налаштовано один пошуковий домен `lpi.org`. Це означає, що будь-яка спроба розпізнати ім'я хоста без доменної частини матиме `.lpi.org`, доданий перед пошуком. Наприклад, якщо ви спробуєте знайти хост під назвою `learning`, резолвер шукатиме `learning.lpi.org`. Ви можете налаштувати до шести доменів пошуку.

Іншим поширеним варіантом є параметр `domain`. Він використовується для встановлення вашого локального доменного імені. Якщо цей параметр відсутній, за замовчуванням буде використано все після першої `.` в імені хоста машини. Якщо ім'я хоста не містить `.`, передбачається, що машина є частиною кореневого домену. Як і `search`, `domain` можна використовувати для пошуку за короткими іменами.

Майте на увазі, що domain` і `search` взаємовиключні. Якщо обидва присутні, використовується останній екземпляр у файлі.

Є кілька параметрів, які можна встановити, щоб вплинути на поведінку резолвера. Щоб установити їх, використовуйте ключове слово `options`, за яким слідує назва параметра, який потрібно встановити, і, якщо це можна застосувати, `:`, а потім значення. Нижче наведено приклад налаштування параметра тайм-ауту, який означає тривалість часу в секундах, протягом якого резолвер чекатиме на сервер імен, перш ніж відмовитися:

```
option timeout:3
```

Існують інші параметри `resolv.conf`, але ці є найпоширенішими.

Файл `/etc/hosts`

Файл `/etc/hosts` використовується для перетворення імен в IP-адреси і навпаки. Підтримуються як IPv4, так і IPv6. Лівий стовпець – це IP-адреса, решта – імена, пов'язані з цією адресою. Найпоширенішим використанням `/etc/hosts` є хости та адреси, де DNS неможливий, наприклад адреси зворотного зв'язку. У наведеному нижче прикладі визначено IP-адреси критичних компонентів інфраструктури.

Ось реалістичний приклад файлу `/etc/hosts`:

```
127.0.0.1       localhost
```

```
127.0.1.1       proxy
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

10.0.0.1        gateway.lpi.org gateway gw
fd00:ffff::1    gateway.lpi.org gateway gw

10.0.1.53       dns1.lpi.org
fd00:ffff::1:53 dns1.lpi.org
10.0.2.53       dns2.lpi.org
fd00:ffff::2:53 dns2.lpi.org
```

systemd-resolved

Systemd надає службу під назвою `systemd-resolved`. Він забезпечує mDNS, DNS і LLMNR. Під час роботи він прослуховує DNS-запити на `127.0.0.53`. Він *не* забезпечує повноцінний сервер DNS. Будь-які запити DNS, які він отримує, обробляються серверами запитів, налаштованими в `/etc/systemd/resolv.conf` або `/etc/resolv.conf`. Якщо ви бажаєте використовувати цю службу, використовуйте `resolve` для `hosts` у `/etc/nsswitch.conf`. Майте на увазі, що пакет ОС, який містить бібліотеку `systemd-resolved`, може бути не встановлений за замовчуванням.

Інструменти розпізнавання імен

Користувачам Linux доступно багато інструментів для розпізнавання імен. Цей урок охоплює три. Один, `getent`, корисний для того, щоб побачити, як будуть вирішуватися запити в реальному світі. Ще одна команда — `host`, яка корисна для простих запитів DNS. Програма під назвою `dig` корисна для складних операцій DNS, яка може допомогти у вирішенні проблем із сервером DNS.

Команда `getent`

Утиліта `getent` використовується для відображення записів із баз даних служби імен. Вона може отримувати записи з будь-якого джерела, яке можна налаштувати за допомогою `/etc/nsswitch.conf`.

To use `getent`, follow the command with the type of name you wish to resolve and optionally a specific entry to lookup. If you only specify the type of name, `getent` will attempt to display all entries of that data type:

```
$ getent hosts
127.0.0.1       localhost
127.0.1.1       proxy
10.0.1.53       dns1.lpi.org
10.0.2.53       dns2.lpi.org
127.0.0.1       localhost ip6-localhost ip6-loopback
$ getent hosts dns1.lpi.org
fd00:ffff::1:53 dns1.lpi.org
```

Starting with glibc version 2.2.5, you can force `getent` to use a specific data source with the `-s` option. The example below demonstrates this:

```
$ getent -s files hosts learning.lpi.org
::1             learning.lpi.org
$ getent -s dns hosts learning.lpi.org
208.94.166.198  learning.lpi.org
```

Команда `host`

`host` — це проста програма для пошуку записів DNS. Без параметрів, якщо `host` передано тільки ім'я, він повертає набори записів A, AAAA та MX. Якщо надається адреса IPv4 або IPv6, він виводить запис PTR, якщо той доступний:

```
$ host wikipedia.org
wikipedia.org has address 208.80.154.224
wikipedia.org has IPv6 address 2620:0:861:ed1a::1
wikipedia.org mail is handled by 10 mx1001.wikimedia.org.
wikipedia.org mail is handled by 50 mx2001.wikimedia.org.
$ host 208.80.154.224
224.154.80.208.in-addr.arpa domain name pointer text-lb.eqiad.wikimedia.org.
```

Якщо ви шукаєте певний тип запису, ви можете використовувати `host -t`:

```
$ host -t NS lpi.org
lpi.org name server dns1.easydns.com.
lpi.org name server dns3.easydns.ca.
lpi.org name server dns2.easydns.net.
$ host -t SOA lpi.org
lpi.org has SOA record dns1.easydns.com. zone.easydns.com. 1593109612 3600 600 1209600 300
```

`host` також можна використовувати для запиту на певний сервер імен, якщо ви не бажаєте використовувати ті, що містяться в `/etc/resolv.conf`. Просто додайте IP-адресу або ім'я хоста сервера, який ви хочете використовувати, як останній аргумент:

```
$ host -t MX lpi.org dns1.easydns.com
Using domain server:
Name: dns1.easydns.com
Address: 64.68.192.10#53
Aliases:

lpi.org mail is handled by 10 aspmx4.googlemail.com.
lpi.org mail is handled by 10 aspmx2.googlemail.com.
lpi.org mail is handled by 5 alt1.aspmx.l.google.com.
lpi.org mail is handled by 0 aspmx.l.google.com.
lpi.org mail is handled by 10 aspmx5.googlemail.com.
lpi.org mail is handled by 10 aspmx3.googlemail.com.
lpi.org mail is handled by 5 alt2.aspmx.l.google.com.
```

Команда `dig`

Іншим інструментом для запиту DNS-серверів є `dig`. Ця команда набагато докладніша, ніж `host`. За замовчуванням запити `dig` для записів A. Можливо, це надто багатослівно для простого пошуку IP-адреси чи імені хоста. `dig` працюватиме для простих пошуків, але він більше підходить для усунення несправностей конфігурації DNS-сервера:

```
$ dig learning.lpi.org

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> learning.lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63004
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ca7a415be1cec45592b082665ef87f3483b81ddd61063c30 (good)
;; QUESTION SECTION:
;learning.lpi.org.      IN  A

;; ANSWER SECTION:
learning.lpi.org.   600 IN  A   208.94.166.198
```

```
;; AUTHORITY SECTION:
lpi.org.        86400   IN  NS  dns2.easydns.net.
lpi.org.        86400   IN  NS  dns1.easydns.com.
lpi.org.        86400   IN  NS  dns3.easydns.ca.

;; ADDITIONAL SECTION:
dns1.easydns.com.   172682  IN  A   64.68.192.10
dns2.easydns.net.   170226  IN  A   198.41.222.254
dns1.easydns.com.   172682  IN  AAAA    2400:cb00:2049:1::a29f:1835
dns2.easydns.net.   170226  IN  AAAA    2400:cb00:2049:1::c629:defe

;; Query time: 135 msec
;; SERVER: 192.168.1.20#53(192.168.1.20)
;; WHEN: Sun Jun 28 07:29:56 EDT 2020
;; MSG SIZE  rcvd: 266
```

Як бачите, `dig` надає багато інформації. Вихідні дані поділені на розділи. У першому розділі відображається інформація про встановлену версію `dig` та надісланий запит разом із усіма параметрами, які використовуються для команди. Далі відображається інформація про запит і відповідь.

У наступному розділі показано інформацію про використані EDNS-розширення і запит. У прикладі використовується розширення cookie. `dig` шукає запис A для `learning.lpi.org`.

У наступному розділі показано результат запиту. Число у другому стовпці – це TTL ресурса в секундах.

Решта вихідних даних надає інформацію про сервери імен домену, включаючи записи NS для сервера разом із записами A і AAAA серверів у записі NS домену.

Як і при використанні команди `host`, ви можете вказати тип запису за допомогою параметра `-t`:

```
$ dig -t SOA lpi.org

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> -t SOA lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16695
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 6

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
```

```
; COOKIE: 185c67140a63baf46c4493215ef8906f7bfbe15bdca3b01a (good)
;; QUESTION SECTION:
;lpi.org.           IN  SOA

;; ANSWER SECTION:
lpi.org.        600 IN  SOA dns1.easydns.com. zone.easydns.com. 1593109612 3600 600 1209600
300

;; AUTHORITY SECTION:
lpi.org.        81989   IN  NS  dns1.easydns.com.
lpi.org.        81989   IN  NS  dns2.easydns.net.
lpi.org.        81989   IN  NS  dns3.easydns.ca.

;; ADDITIONAL SECTION:
dns1.easydns.com.   168271  IN  A   64.68.192.10
dns2.easydns.net.   165815  IN  A   198.41.222.254
dns3.easydns.ca.    107 IN  A   64.68.196.10
dns1.easydns.com.   168271  IN  AAAA    2400:cb00:2049:1::a29f:1835
dns2.easydns.net.   165815  IN  AAAA    2400:cb00:2049:1::c629:defe

;; Query time: 94 msec
;; SERVER: 192.168.1.20#53(192.168.1.20)
;; WHEN: Sun Jun 28 08:43:27 EDT 2020
;; MSG SIZE  rcvd: 298
```

Dig має багато параметрів для точного налаштування виведення та запиту, надісланого на сервер. Ці параметри починаються з `+`. Одним із варіантів є параметр `short`, який приховує всі виведення, крім результату:

```
$ dig +short lpi.org
65.39.134.165
$ dig +short -t SOA lpi.org
dns1.easydns.com. zone.easydns.com. 1593109612 3600 600 1209600 300
```

Ось приклад вимкнення розширення cookie EDNS:

```
$ dig +nocookie -t MX lpi.org

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> +nocookie -t MX lpi.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47774
```

```
;; flags: qr rd ra; QUERY: 1, ANSWER: 7, AUTHORITY: 3, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;lpi.org.           IN  MX

;; ANSWER SECTION:
lpi.org.        468 IN  MX  0 aspmx.l.google.com.
lpi.org.        468 IN  MX  10 aspmx4.googlemail.com.
lpi.org.        468 IN  MX  10 aspmx5.googlemail.com.
lpi.org.        468 IN  MX  10 aspmx2.googlemail.com.
lpi.org.        468 IN  MX  10 aspmx3.googlemail.com.
lpi.org.        468 IN  MX  5 alt2.aspmx.l.google.com.
lpi.org.        468 IN  MX  5 alt1.aspmx.l.google.com.

;; AUTHORITY SECTION:
lpi.org.        77130   IN  NS  dns2.easydns.net.
lpi.org.        77130   IN  NS  dns3.easydns.ca.
lpi.org.        77130   IN  NS  dns1.easydns.com.

;; ADDITIONAL SECTION:
dns1.easydns.com.   76140   IN  A   64.68.192.10
dns2.easydns.net.   73684   IN  A   198.41.222.254
dns1.easydns.com.   76140   IN  AAAA    2400:cb00:2049:1::a29f:1835
dns2.easydns.net.   73684   IN  AAAA    2400:cb00:2049:1::c629:defe

;; Query time: 2 msec
;; SERVER: 192.168.1.20#53(192.168.1.20)
;; WHEN: Mon Jun 29 10:18:58 EDT 2020
;; MSG SIZE  rcvd: 389
```

Вправи до посібника

1. Що буде робити команда нижче?

```
getent group openldap
```

2. Яка найбільша різниця між `getent` та іншими розглянутими інструментами, `host` і `dig`?

3. Який параметр `dig` і `host` використовується для визначення типу запису, який ви бажаєте отримати?

4. Що з наведеного нижче є правильним записом `/etc/hosts`?

`::1 localhost`	
`localhost 127.0.0.1`	

5. Який параметр `getent` використовується для визначення джерела даних, яке слід використовувати для виконання пошуку?

Дослідницькі вправи

1. Якби ви редагували `/etc/resolv.conf` нижче за допомогою текстового редактора, що, ймовірно, трапиться?

```
# Generated by NetworkManager
nameserver 192.168.1.20
```

Зміни буде перезаписано NetworkManager.	
NetworkManager оновить свою конфігурацію відповідно до ваших змін.	
Ваші зміни не вплинуть на систему.	
NetworkManager буде вимкнено.	

2. Що означає наступний рядок у `/etc/nsswitch.conf`:

```
hosts: files [SUCCESS=continue] dns
```

3. Враховуючи наступний `/etc/resolv.conf`, чому система не розпізнає імена через DNS?

```
search lpi.org
#nameserver fd00:ffff::1:53
#nameserver 10.0.1.53
```

4. Що робить команда `dig +noall +answer +question lpi.org`?

5. Як можна перевизначити значення за замовчуванням `dig`, не вказавши їх у командному рядку?

Підсумки

Команда `getent` є чудовим інструментом для перегляду результатів викликів резолвера. Для простих DNS-запитів `host` простий у використанні та видає прості вихідні дані. Якщо вам потрібна детальна інформація або потрібно точно налаштувати DNS-запит, `dig`, швидше за все, найкращий вибір.

Завдяки можливості додавати модулі спільної бібліотеки та налаштовувати поведінку резолвера, Linux чудово підтримує розпізнавання імен і чисел різних типів. Програму `getent` можна використовувати для розпізнавання імен за допомогою бібліотек розпізнавача. `host` і `dig` можна використовувати для запиту DNS-серверів.

Файл `/etc/nsswitch.conf` використовується для налаштування поведінки резолвера. Ви можете змінювати джерела даних і додавати просту умовну логіку для типів імен із кількома джерелами.

DNS налаштовується шляхом редагування `/etc/resolv.conf`. Багато дистрибутивів мають інструменти, які керують цим файлом за вас, тому перевірте документацію вашої системи, якщо внесені вручну зміни не зберігаються.

Файл `/etc/hosts` використовується для визначення імен хостів відповідно до IP-адреси і навпаки. Зазвичай він використовується для визначення імен, таких як `localhost`, які недоступні через DNS.

Можна залишати коментарі у конфігураційних файлах, розглянутих у цьому уроці. Будь-який текст праворуч від `#` ігнорується системою.

Відповіді до вправ посібника

1. Що буде робити команда нижче?

```
getent group openldap
```

   Вона читатиме `/etc/nsswitch.conf`, шукатиме групу `openldap` із перелічених джерел і відображатиме інформацію про неї, якщо її знайдено.

2. Яка найбільша різниця між `getent` та іншими розглянутими інструментами, `host` і `dig`?

   `getent` шукає імена за допомогою бібліотек резолвера, інші просто запитують DNS. `getent` можна використовувати для усунення несправностей вашого `/etc/nsswitch.conf` і конфігурації бібліотек розпізнавання імен, налаштованих на використання вашою системою. `host` і `dig` використовуються для пошуку записів DNS.

3. Який параметр `dig` і `host` використовується для визначення типу запису, який ви бажаєте отримати?

   Обидві програми використовують `-t`, щоб вказати тип запису, який ви бажаєте знайти.

4. Що з наведеного нижче є правильним записом `/etc/hosts`?

`::1 localhost`	X
`localhost 127.0.0.1`	

   `::1 localhost` правильний рядок. Ліва колонка завжди містить адресу IPv4 або IPv6.

5. Який параметр `getent` використовується для вказівки джерела даних, яке слід використовувати для виконання пошуку?

   Параметр `-s` використовується для визначення джерела даних. Наприклад:

```
$ getent -s files hosts learning.lpi.org
192.168.10.25   learning.lpi.org
$ getent -s dns hosts learning.lpi.org
208.94.166.198  learning.lpi.org
```

Відповіді до дослідницьких вправ

1. Якби ви редагували `/etc/resolv.conf` нижче за допомогою текстового редактора, що, ймовірно, трапиться?

```
# Generated by NetworkManager
nameserver 192.168.1.20
```

Зміни буде перезаписано NetworkManager.	X
NetworkManager оновить свою конфігурацію відповідно до ваших змін.	
Ваші зміни не вплинуть на систему.	
NetworkManager буде вимкнено.	

2. Що означає наступний рядок у `/etc/nsswitch.conf`:

```
hosts: files [SUCCESS=continue] dns
```

   Пошук імен хостів спочатку перевірить ваші файли `/etc/hosts`, а потім DNS. Якщо шуканий запис знайдено у файлах і DNS, буде використано запис у DNS.

3. Враховуючи наступний `/etc/resolv.conf`, чому система не розпізнає імена через DNS?

```
search lpi.org
#nameserver fd00:ffff::1:53
#nameserver 10.0.1.53
```

   Обидва DNS-сервери закоментовані, і на локальному хості не працює DNS-сервер.

4. Що робить команда `dig +noall +answer +question lpi.org`?

   Він шукає запис A для lpi.org і відображає лише запит і відповідь.

5. Як можна перевизначити значення за замовчуванням `dig`, не вказавши їх у командному рядку?

   Ви створюєте файл `.digrc` у своєму домашньому каталозі.

Linux Professional Institute

Розділ 110: Безпека

110.1 Виконання завдань з адміністрування безпеки

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 110.1

Ваговий коефіцієнт

3

Ключові галузі знань

- Аудит системи для пошуку файлів із встановленим бітом suid/sgid.
- Встановлення або зміна паролів користувачів і інформації про закінчення терміну дії паролів.
- Можливість використовувати nmap і netstat для виявлення відкритих портів у системі.
- Встановлення обмеження на вхід користувачів, процеси та використання пам’яті.
- Визначення, які користувачі ввійшли в систему або в даний момент ввійшли в систему.
- Базова конфігурація та використання sudo.

Частковий список файлів, термінів та утиліт, що використовуються

- `find`
- `passwd`
- `fuser`
- `lsof`
- `nmap`
- `chage`

- `netstat`
- `sudo`
- `/etc/sudoers`
- `su`
- `usermod`
- `ulimit`
- `who`, `w`, `last`

110.1 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	110 Безпека
Тема:	110.1 Виконання завдань з адміністрування безпеки
Урок:	1 з 1

Вступ

Безпека є обов'язковою умовою адміністрування системи. Як хороший системний адміністратор Linux, ви повинні стежити за низкою речей, таких як спеціальні дозволи на файли, застарілі паролі користувачів, відкриті порти та сокети, обмеження використання системних ресурсів, робота з користувачами, які ввійшли в систему, і підвищення привілеїв за допомогою `su` і `sudo`. На цьому уроці ми розглянемо кожну з цих тем.

Перевірка файлів із набором SUID і SGID

Крім традиційного набору дозволів *read*, *write* і *execute*, файли в системі Linux також можуть мати деякі спеціальні дозволи, такі як *SUID* або *SGID*-біти.

SUID-біт дозволить запустити файл із правами власника файлу. У числовому форматі він представлений як `4000` і символьно представлений `s` або `S` у біті дозволу *execute* власника. Класичним прикладом виконуваного файлу з набором SUID-дозволів є `passwd`:

```
carol@debian:~$ ls -l /usr/bin/passwd
```

```
-rwsr-xr-x 1 root root 63736 jul 27  2018 /usr/bin/passwd
```

`s` у нижньому регістрі в `rws` вказує на наявність SUID у файлі разом із дозволом *execute*. Велике `S` натомість (`rwS`) означає, що основний дозвіл *execute* не встановлено.

NOTE

Ви дізнаєтесь про `passwd` у наступному розділі. Утиліта здебільшого використовується `root` для встановлення/зміни паролів користувачів (наприклад: `passwd carol`). Однак звичайні користувачі також можуть використовувати його для зміни власних паролів. Тому команда має опцію SUID.

З іншого боку, SGID-біт можна встановити як для файлів, так і для каталогів. З файлами його поведінка еквівалентна SUID, але привілеї належать власнику групи. Однак, якщо його встановити для каталогу, він дозволить усім створеним у ньому файлам успадкувати право власності на групу каталогу. Подібно до SUID, SGID символьно представлено або `s`, або `S` у біті дозволу *execute* групи. У числовому форматі він представлений як `2000`. Ви можете встановити SGID для каталогу за допомогою `chmod`. Ви повинні додати `2` (SGID) до традиційних дозволів (`755` у нашому випадку):

```
carol@debian:~$ ls -ld shared_directory
drwxr-xr-x 2 carol carol 4096 may 30 23:55 shared_directory
carol@debian:~$ sudo chmod 2755 shared_directory/
carol@debian:~$ ls -ld shared_directory
drwxr-sr-x 2 carol carol 4096 may 30 23:55 shared_directory
```

Щоб знайти файли зі встановленим SUID чи SGID або з обома, ви можете скористатися командою `find` і опцією `-perm`. Ви можете використовувати як числові, так і символьні значення. Значення, у свою чергу, можуть передаватися окремо або перед ними ставитися тире (`-`) або коса риска (`/`). Суть полягає в наступному:

`-perm numeric-value` або `-perm symbolic-value`
: знайти файли *виключно* зі спеціальним дозволом

`-perm -numeric-value` або `-perm -symbolic-value`
: знайти файли зі спеціальним дозволом та іншими дозволами

`-perm /numeric-value` або `-perm /symbolic-value`
: знайти файли зі спеціальним дозволом (та іншими дозволами)

Наприклад, щоб знайти файли з *тільки* SUID, встановленим у поточному робочому

каталозі, ви скористаєтеся такою командою:

```
carol@debian:~$ find . -perm 4000
carol@debian:~$ touch file
carol@debian:~$ chmod 4000 file
carol@debian:~$ find . -perm 4000
./file
```

Зауважте, що, оскільки файлів для яких встановлено тільки SUID не було, ми створили файл, щоб показати яким буде результат. Ви можете виконати цю ж команду в символьному позначенні:

```
carol@debian:~$ find . -perm u+s
./file
```

Щоб знайти файли, що відповідають SUID (незалежно від будь-яких інших дозволів) у каталозі `/usr/bin/`, ви можете використати одну з наступних команд:

```
carol@debian:~$ sudo find /usr/bin -perm -4000
/usr/bin/umount
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/mount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/su
carol@debian:~$ sudo find /usr/bin -perm -u+s
/usr/bin/umount
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/mount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/su
```

Якщо ви шукаєте файли в одному каталозі з установленим SGID-бітом, ви можете виконати `find /usr/bin/ -perm -2000` або `find /usr/bin/ -perm -g+s`.

Нарешті, щоб знайти файли з будь-яким із двох спеціальних дозволів, додайте `4` і `2` і використовуйте `/`:

```
carol@debian:~$ sudo find /usr/bin -perm /6000
/usr/bin/dotlock.mailutils
/usr/bin/umount
/usr/bin/newgrp
/usr/bin/wall
/usr/bin/ssh-agent
/usr/bin/chage
/usr/bin/dotlockfile
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/mount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/expiry
/usr/bin/sudo
/usr/bin/bsd-write
/usr/bin/crontab
/usr/bin/su
```

Керування паролями та старіння

Як зазначено вище, ви можете використовувати утиліту `passwd`, щоб змінити свій власний пароль як звичайний користувач. Крім того, ви можете передати опцію `-S` або `--status`, щоб отримати інформацію про статус вашого облікового запису:

```
carol@debian:~$ passwd -S
carol P 12/07/2019 0 99999 7 -1
```

Ось розбивка семи полів, які ви отримуєте у вихідних даних:

`carol`

Ім’я користувача для входу.

`P`

Вказує, що користувач має дійсний пароль (`P`); інші можливі значення: `L` для заблокованого пароля та `NP` для відсутності пароля.

`07.12.2019`

Дата останньої зміни пароля.

`0`

Мінімальний вік у днях (мінімальна кількість днів між змінами пароля). Значення `0` означає, що пароль можна змінити в будь-який час.

`99999`

Максимальний вік у днях (максимальна кількість днів, протягом яких пароль дійсний). Значення `99999` вимкне термін дії пароля.

`7`

Період попередження в днях (кількість днів до закінчення терміну дії пароля, коли користувач отримає попередження).

`-1`

Період неактивності пароля в днях (кількість неактивних днів після закінчення терміну дії пароля до блокування облікового запису). Значення `-1` видалить неактивний обліковий запис.

Окрім звітування про стан облікового запису, ви використовуватимете команду `passwd` як користувач root, щоб здійснювати базове обслуговування облікового запису. Ви можете заблокувати та розблокувати облікові записи, змусити користувача змінити свій пароль під час наступного входу та видалити пароль користувача за допомогою опцій `-l`, `-u`, `-e` та `-d` відповідно.

Щоб перевірити ці параметри, тут зручно ввести команду `su`. За допомогою `su` ви можете змінити користувачів під час сеансу входу. Отже, наприклад, використаємо `passwd` як root, щоб заблокувати пароль для `carol`. Потім ми зайдемо як `carol` і перевіримо стан нашого облікового запису, щоб переконатися, що пароль — насправді — заблоковано (`L`) і його неможливо змінити. Нарешті, повернувшись до користувача root, ми розблокуємо пароль для `carol`:

```
root@debian:~# passwd -l carol
passwd: password expiry information changed.
root@debian:~# su - carol
carol@debian:~$ passwd -S
carol L 05/31/2020 0 99999 7 -1
carol@debian:~$ passwd
Changing password for carol.
Current password:
```

```
passwd: Authentication token manipulation error
passwd: password unchanged
carol@debian:~$ exit
logout
root@debian:~# passwd -u carol
passwd: password expiry information changed.
```

Крім того, ви також можете заблокувати та розблокувати пароль користувача за допомогою команди `usermod`:

Заблокувати пароль користувача `carol`

`usermod -L carol` або `usermod --lock carol`.

Розблокувати пароль користувача `carol`

`usermod -U carol` або `usermod --unlock carol`.

NOTE За допомогою перемикачів `-f` або `--inactive` `usermod` також можна встановити кількість днів до вимкнення облікового запису з паролем, термін якого сплив (наприклад: `usermod -f 3 carol`).

Окрім `passwd` і `usermod`, найпрямішою командою для роботи з паролем і старінням облікового запису є `chage` (“змінити вік”). Як root ви можете передати `chage` перемикач `-l` (або `--list`), а потім ім’я користувача, щоб на екрані виводилися поточний пароль користувача та інформація про закінчення терміну дії облікового запису; як звичайний користувач, ви можете переглядати власну інформацію:

```
carol@debian:~$ chage -l carol
Last password change                    : Aug 06, 2019
Password expires                        : never
Password inactive                       : never
Account expires                         : never
Minimum number of days between password change      : 0
Maximum number of days between password change      : 99999
Number of days of warning before password expires   : 7
```

Запустіть команду `chage` без параметрів і лише після імені користувача. Тоді вона поводитиметься інтерактивно:

```
root@debian:~# chage carol
Changing the aging information for carol
Enter the new value, or press ENTER for the default
```

```
Minimum Password Age [0]:
Maximum Password Age [99999]:
Last Password Change (YYYY-MM-DD) [2020-06-01]:
Password Expiration Warning [7]:
Password Inactive [-1]:
Account Expiration Date (YYYY-MM-DD) [-1]:
```

Варіанти зміни різних налаштувань `chage` є такими:

`-m days username` або `--mindays days username`

Вкажіть мінімальну кількість днів між змінами пароля (наприклад: `chage -m 5 carol`). Значення `0` дозволить користувачеві змінити свій пароль у будь-який час.

`-M days username` або `--maxdays days username`

Вкажіть максимальну кількість днів, протягом яких пароль буде дійсним (наприклад: `chage -M 30 carol`). Щоб вимкнути термін дії пароля, прийнято надавати цій опції значення `99999`.

`-d days username` або `--lastday days username`

Вкажіть кількість днів після останньої зміни пароля (наприклад: `chage -d 10 carol`). Значення `0` змусить користувача змінити свій пароль під час наступного входу.

`-W days username` або `--warndays days username`

Вкажіть кількість днів, протягом яких користувачеві буде нагадано про закінчення терміну дії пароля.

`-I days username` або `--inactive days username`

Вкажіть кількість неактивних днів після закінчення терміну дії пароля (наприклад: `chage -I 10 carol`) — те саме, що `usermod -f` або `usermod --inactive`. Коли ця кількість днів мине, обліковий запис буде заблоковано. Проте зі значенням `0` обліковий запис не буде заблоковано.

`-E date username` або `--expiredate date username`

Вкажіть дату (або кількість днів з *епохи* — 1 січня 1970 року), коли обліковий запис буде заблоковано. Зазвичай він виражається у форматі `YYYY-MM-DD` (наприклад: `chage -E 2050-12-13 carol`).

NOTE Ви можете дізнатися більше про `passwd`, `usermod` і `chage` — та їх параметри — звернувшись до відповідних man-сторінок.

Виявлення відкритих портів

Коли справа доходить до спостереження за відкритими портами, у більшості систем Linux присутні чотири потужні утиліти: `lsof`, `fuser`, `netstat` і `nmap`. Ми розглянемо їх у цьому розділі.

`lsof` означає “список відкритих файлів”, що не дрібниця, враховуючи, що для Linux все є файлом. Насправді, якщо ви введете `lsof` у терміналі, ви отримаєте великий список звичайних файлів, файлів пристроїв, сокетів тощо. Однак для цього уроку ми зосередимося переважно на портах. Щоб надрукувати список усіх мережних файлів “Internet”, запустіть `lsof` з опцією `-i`:

```
root@debian:~# lsof -i
COMMAND  PID     USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
dhclient 357     root    7u  IPv4  13493      0t0  UDP *:bootpc
sshd     389     root    3u  IPv4  13689      0t0  TCP *:ssh (LISTEN)
sshd     389     root    4u  IPv6  13700      0t0  TCP *:ssh (LISTEN)
apache2  399     root    3u  IPv6  13826      0t0  TCP *:http (LISTEN)
apache2  401 www-data    3u  IPv6  13826      0t0  TCP *:http (LISTEN)
apache2  402 www-data    3u  IPv6  13826      0t0  TCP *:http (LISTEN)
sshd     557     root    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510
(ESTABLISHED)
sshd     569    carol    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510
(ESTABLISHED)
```

Окрім служби `bootpc`, яка використовується DHCP, вихідні дані показують дві служби, які прослуховують з’єднання — `ssh` і веб-сервер Apache (`http`), а також два встановлені з’єднання SSH. Ви можете вказати певний хост за допомогою запису `@ip-address`, щоб перевірити наявність його з’єднань:

```
root@debian:~# lsof -i@192.168.1.7
COMMAND PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    557  root    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510
(ESTABLISHED)
sshd    569 carol    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510
(ESTABLISHED)
```

NOTE Щоб вивести лише мережні файли IPv4 та IPv6, використовуйте параметри `-i4` та `-i6` відповідно.

Так само ви можете фільтрувати за портом, передавши опцію `-i` (або `-i@ip-address`)

аргументу `:port`:

```
root@debian:~# lsof -i :22
COMMAND PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    389  root    3u  IPv4  13689      0t0  TCP *:ssh (LISTEN)
sshd    389  root    4u  IPv6  13700      0t0  TCP *:ssh (LISTEN)
sshd    557  root    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510
(ESTABLISHED)
sshd    569 carol    3u  IPv4  14701      0t0  TCP 192.168.1.7:ssh->192.168.1.4:60510
(ESTABLISHED)
```

Кілька портів розділяються комами (діапазони вказуються через тире):

```
root@debian:~# lsof -i@192.168.1.7:22,80
COMMAND PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    705  root    3u  IPv4  13960      0t0  TCP 192.168.1.7:ssh->192.168.1.4:44766
(ESTABLISHED)
sshd    718 carol    3u  IPv4  13960      0t0  TCP 192.168.1.7:ssh->192.168.1.4:44766
(ESTABLISHED)
```

NOTE Кількість опцій, доступних для `lsof`, досить вражаюча. Щоб дізнатися більше, зверніться до man-сторінки.

Наступною у списку мережних команд є `fuser`. Її головна мета полягає в тому, щоб знайти “користувача файлу”, що передбачає знання того, які процеси отримують доступ до яких файлів; вона також надає іншу інформацію, наприклад тип доступу. Наприклад, щоб перевірити поточний робочий каталог, достатньо запустити `fuser`. Однак, щоб отримати трохи більше інформації, зручно використовувати параметр verbose (`-v` або `--verbose`):

```
root@debian:~# fuser .
/root:                580c
root@debian:~# fuser -v .
                     USER        PID ACCESS COMMAND
/root:               root        580 ..c.. bash
```

Розберемо результат:

File

Файл, про який ми отримуємо інформацію (`/root`).

Стовпець `USER`

власник файлу (`root`). Стовпець `PID`: Ідентифікатор процесу (`580`).

Стовпець `ACCESS`

Тип доступу (`..c..`). Один з:

`c`

Поточний каталог.

`e`

Запускається виконуваний файл.

`f`

Відкрити файл (пропущено в режимі відображення за замовчуванням).

`F`

Відкрити файл для запису (пропущено в режимі відображення за замовчуванням).

`r`

Кореневий каталог.

`m`

mmap-файл або спільна бібліотека.

`.`

Заповнювач (пропущено в режимі відображення за замовчуванням).

Стовпець `COMMAND`

Команда, пов'язана з файлом (`bash`).

За допомогою параметра `-n` (або `--namespace`) ви можете знайти інформацію про мережні порти/сокети. Ви також повинні вказати мережний протокол і номер порту. Таким чином, щоб отримати інформацію про веб-сервер Apache, ви запустите таку команду:

```
root@debian:~# fuser -vn tcp 80
                     USER        PID ACCESS COMMAND
80/tcp:              root        402 F.... apache2
                     www-data    404 F.... apache2
                     www-data    405 F.... apache2
```

NOTE

`fuser` також можна використовувати для припинення процесів, які звертаються до файлу за допомогою перемикачів `-k` або `--kill` (наприклад: `fuser -k 80/tcp`). Зверніться до man-сторінки для отримання більш детальної інформації.

Тепер перейдемо до `netstat`. `netstat` — це дуже універсальний мережевий інструмент, який переважно використовується для виведення "мережної статистики".

Без опцій `netstat` відображатиме як активні підключення до Інтернету, так і сокети Unix. Через розмір виведеного списку ви можете захотіти передати його вихідні дані через `less`:

```
carol@debian:~$ netstat |less
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.1.7:ssh         192.168.1.4:55444       ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    10509    /run/systemd/journal/syslog
unix  3      [ ]         DGRAM                    10123    /run/systemd/notify
(...)
```

Щоб вивести лише порти та сокети, які "прослуховуються", використовуйте параметри `-l` або `--listening`. Параметри `-t`/`--tcp` і `-u`/`--udp` можна додати для фільтрації за протоколом TCP і UDP відповідно (їх також можна поєднати в одній команді). Так само `-e`/`--extend` покаже додаткову інформацію:

```
carol@debian:~$ netstat -lu
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
carol@debian:~$ netstat -lt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 localhost:smtp          0.0.0.0:*               LISTEN
tcp6       0      0 [::]:http               [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
tcp6       0      0 localhost:smtp          [::]:*                  LISTEN
carol@debian:~$ netstat -lute
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User
Inode
```

```
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN      root
13729
tcp        0      0 localhost:smtp          0.0.0.0:*               LISTEN      root
14372
tcp6       0      0 [::]:http               [::]:*                  LISTEN      root
14159
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN      root
13740
tcp6       0      0 localhost:smtp          [::]:*                  LISTEN      root
14374
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*                           root
13604
```

Якщо ви не вкажете опцію `-l`, будуть показані *лише* встановлені з'єднання:

```
carol@debian:~$ netstat -ute
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User
Inode
tcp        0      0 192.168.1.7:ssh         192.168.1.4:39144       ESTABLISHED root
15103
```

Якщо вас цікавить лише числова інформація про порти та хости, ви можете використати опцію `-n` або `--numeric`, щоб вивести лише номери портів та IP-адреси. Зверніть увагу, як `ssh` перетворюється на `22` при додаванні `-n` до команди вище:

```
carol@debian:~$ netstat -uten
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       User
Inode
tcp        0      0 192.168.1.7:22          192.168.1.4:39144       ESTABLISHED 0
15103
```

Як можете бачити, ви можете створити дуже корисні та продуктивні команди `netstat`, комбінуючи деякі з її параметрів. Перегляньте man-сторінки, щоб дізнатися більше та знайти комбінації, які найкраще відповідають вашим потребам.

Нарешті, ми представимо `nmap` — або "network mapper". Ще одна дуже потужна утиліта, цей сканер портів виконується шляхом зазначення IP-адреси або імені хоста:

```
root@debian:~# nmap localhost
```

```
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 19:29 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000040s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 1.58 seconds
```

Окрім одного хосту, `nmap` дозволяє сканувати:

декілька хостів

розділивши їх пробілами(наприклад, `nmap localhost 192.168.1.7`).

діапазони хостів

за допомогою тире (наприклад, `nmap 192.168.1.3-20`).

підмережі

за допомогою символу підстановки або нотації CIDR (наприклад, `nmap 192.168.1.*` або `nmap 192.168.1.0/24`). Ви можете виключити окремі хости(наприклад, `nmap 192.168.1.0/24 --exclude 192.168.1.7`).

Щоб просканувати певний порт, скористайтеся опцією `-p`, а потім номером порту або назвою служби (`nmap -p 22` і `nmap -p ssh` дадуть той самий результат):

```
root@debian:~# nmap -p 22 localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 19:54 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000024s latency).
Other addresses for localhost (not scanned): ::1

PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
```

Ви також можете сканувати кілька портів або діапазонів портів, використовуючи коми та тире відповідно:

```
root@debian:~# nmap -p ssh,80 localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 19:58 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000051s latency).
Other addresses for localhost (not scanned): ::1

PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
```

```
root@debian:~# nmap -p 22-80 localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2020-06-04 19:58 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000011s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 57 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 1.47 seconds
```

Ще два важливі та зручні параметри `nmap`:

`-F`
: запустити швидке сканування 100 найпоширеніших портів.

`-v`
: отримати докладне виведення (`-vv` виведе ще більш докладний результат).

NOTE `nmap` може виконувати досить складні команди, використовуючи типи сканування. Однак ця тема виходить за рамки цього уроку.

Обмеження на логіни користувачів, процеси та використання пам’яті

Ресурси в системі Linux не безмежні, тому як системний адміністратор ви повинні забезпечити хороший баланс між *обмеженнями* користувачів на ресурси та належним

функціонуванням операційної системи. `ulimit` може допомогти вам у цьому.

`ulimit` має справу з *м'якими* і *жорсткими* обмеженнями, визначеними параметрами `-S` і `-H` відповідно. Без параметрів або аргументів `ulimit` відобразить блоки програмного файлу поточного користувача:

```
carol@debian:~$ ulimit
unlimited
```

З опцією `-a` `ulimit` покаже всі поточні м'які обмеження (те саме, що `-Sa`); щоб відобразити всі поточні жорсткі обмеження, використовуйте `-Ha`:

```
carol@debian:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
(...)
carol@debian:~$ ulimit -Ha
core file size          (blocks, -c) unlimited
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
(...)
```

Доступні ресурси оболонки визначаються такими параметрами, як:

`-b`

максимальний розмір буфера сокета

`-f`

максимальний розмір файлів, записаних оболонкою та її дочірніми елементами

`-l`

максимальний розмір, який можна заблокувати в пам'яті

`-m`

максимальний розмір резидентного набору (RSS) — поточна частина пам'яті, яку зберігає процес в основній пам'яті (RAM)

`-v`

максимальний обсяг віртуальної пам'яті

`-u`

максимальна кількість процесів, доступних для одного користувача

Таким чином, щоб відобразити обмеження, ви будете використовувати `ulimit`, за яким слідує опція `-S` (м'який) або `-H` (жорсткий) і параметр ресурсу; якщо не вказано ні `-S`, ні `-H`, будуть показані м'які обмеження:

```
carol@debian:~$ ulimit -u
10000
carol@debian:~$ ulimit -Su
10000
carol@debian:~$ ulimit -Hu
15672
```

Подібним чином, щоб установити нові обмеження для певного ресурсу, ви вкажете `-S` або `-H`, а потім відповідний параметр ресурсу та нове значення. Це значення може бути числом або спеціальними словами `soft` (поточний м'який ліміт), `hard` (поточний жорсткий ліміт) або `unlimited` (без обмеження). Якщо не вказано ні `-S`, ні `-H`, буде встановлено обидва обмеження. Наприклад, давайте спочатку прочитаємо поточне значення максимального розміру для файлів, написаних оболонкою та її дочірніми елементами:

```
root@debian:~# ulimit -Sf
unlimited
root@debian:~# ulimit -Hf
unlimited
```

Тепер давайте змінимо значення з `unlimited` на `500` блоків, не вказуючи ні `-S`, ні `-H`. Зверніть увагу, як змінюються м'які та жорсткі обмеження:

```
root@debian:~# ulimit -f 500
root@debian:~# ulimit -Sf
500
root@debian:~# ulimit -Hf
500
```

Нарешті, ми зменшимо лише м'яке обмеження до `200` блоків:

```
root@debian:~# ulimit -Sf 200
```

```
root@debian:~# ulimit -Sf
200
root@debian:~# ulimit -Hf
500
```

Жорсткі обмеження може збільшити лише користувач root. З іншого боку, звичайні користувачі можуть зменшувати жорсткі обмеження та збільшувати м'які обмеження до значення жорстких обмежень. Щоб зробити нові граничні значення постійними під час перезавантаження, ви повинні записати їх у файл `/etc/security/limits.conf`. Цей файл також використовується адміністратором для застосування обмежень для окремих користувачів.

NOTE Майте на увазі, що не існує man-сторінки `ulimit` як такої. Це вбудований bash, тому вам потрібно звернутися до man-сторінки bash, щоб дізнатися про це.

Робота з зареєстрованими користувачами

Інша ваша робота як системного адміністратора передбачає відстеження користувачів, які ввійшли в систему. Є три утиліти, які можуть допомогти вам із цими завданнями: `last`, `who` і `w`.

`last` виводить список останніх користувачів, які входили в систему, з найновішою інформацією вгорі:

```
root@debian:~# last
carol    pts/0        192.168.1.4      Sat Jun  6 14:25   still logged in
reboot   system boot  4.19.0-9-amd64   Sat Jun  6 14:24   still running
mimi     pts/0        192.168.1.4      Sat Jun  6 12:07 - 14:24  (02:16)
reboot   system boot  4.19.0-9-amd64   Sat Jun  6 12:07 - 14:24  (02:17)
(...)
wtmp begins Sun May 31 14:14:58 2020
```

Враховуючи скорочений лістинг, ми отримуємо інформацію про двох останніх користувачів у системі. Перші два рядки розповідають нам про користувача `carol`; наступні два рядки про користувача `mimi`. Інформація така:

1. Користувач `carol` на терміналі `pts/0` з хосту `192.168.1.4` розпочав свій сеанс `Sat Jun 6` о `14:25` і все ще `знаходиться в ній (logged in)`. Система з використанням ядра `4.19.0-9-amd64` була запущена (`reboot system boot`) `Sat Jun 6` о 14:24` і `досі працює (still running)`.

2. Користувач `mimi` на терміналі `pts/0` з хоста `192.168.1.4` розпочала свою сесію `Sat Jun 6 в 12:07` і закінчила об `14:24` (сесія тривала загалом `(02:16)` години). Система — використовуючи ядро `4.19.0-9-amd64` — запустилася (`reboot system boot`) `Sat Jun 6` в `12:07` і вимкнулася `14:24` (загалом вона працювала `(02:17)` години).

> **NOTE** Рядок `wtmp begins Sun May 31 14:14:58 2020` посилається на `/var/log/wtmp`, який є спеціальним файлом журналу, з якого `last` отримує інформацію.

Ви можете передати команді `last` ім'я користувача, щоб відображалися лише записи для цього користувача:

```
root@debian:~# last carol
carol    pts/0        192.168.1.4      Sat Jun  6 14:25   still logged in
carol    pts/0        192.168.1.4      Sat Jun  6 12:07 - 14:24  (02:16)
carol    pts/0        192.168.1.4      Fri Jun  5 00:48 - 01:28  (00:39)
(...)
```

Щодо другого стовпця (терміналу), `pts` означає *Pseudo Terminal Slave* — на відміну від *TeleTYpewriter* або `tty` терміналу; `0` відноситься до першого (рахунок починається з нуля).

> **NOTE** Щоб перевірити помилкові спроби входу, запустіть `lastb` замість `last`.

Утиліти `who` і `w` зосереджені на користувачах, які зараз увійшли в систему, і вони досить схожі. Перша показує, хто ввійшов, а друга також показує інформацію про те, що вони роблять.

Без параметрів `who` відобразить чотири стовпці, що відповідають користувачу, який увійшов у систему, терміналу, даті й часу та імені хоста:

```
root@debian:~# who
carol    pts/0        2020-06-06 17:16 (192.168.1.4)
mimi     pts/1        2020-06-06 17:28 (192.168.1.4)
```

`who` приймає низку опцій, серед яких можна виділити наступні:

`-b`, **`--boot`**

Показати час останнього завантаження системи.

`-r`, **`--runlevel`**

Показати поточний рівень виконання.

`-H`, `--heading`

Друкує заголовки стовпців.

Порівняно з `who`, `w` дає трохи докладнішу інформацію:

```
root@debian:~# w
 17:56:12 up 40 min,  2 users,  load average: 0.04, 0.12, 0.09
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
carol    pts/0    192.168.1.4    17:16    1.00s  0.15s  0.05s sshd: carol [priv]
mimi     pts/1    192.168.1.4    17:28   15:08   0.05s  0.05s -bash
```

У верхньому рядку надається інформація про поточний час (`17:56:12`), як довго система працює (`up 40 min`), кількість користувачів, які зараз увійшли в систему (`2 users`) і значення середнього навантаження (`load average: 0.04, 0.12, 0.09`). Ці значення відносяться до кількості завдань у черзі виконання, усередненої за останні 1, 5 і 15 хвилин відповідно.

Потім ви знайдете вісім колонок; давайте розберемо їх:

`USER`

Логін користувача.

`TTY`

Назва терміналу, на якому перебуває користувач.

`FROM`

Віддалений хост, з якого користувач увійшов.

`LOGIN@`

Час входу.

`IDLE`

Час простою.

`JCPU`

Час, використаний усіма процесами, підключеними до tty (включно з поточними фоновими завданнями).

`PCPU`

Час, використаний поточним процесом (той, що відображається під `WHAT`).

`WHAT`

Командний рядок поточного процесу.

Подібно до `who`, ви можете передати `w` імена користувачів:

```
root@debian:~# w mimi
 18:23:15 up  1:07,  2 users,  load average: 0.00, 0.02, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
mimi     pts/1    192.168.1.4      17:28    9:23   0.06s  0.06s -bash
```

Базова конфігурація та використання `sudo`

Як уже зазначалося в цьому уроці, `su` дозволяє вам переключитися на будь-якого іншого користувача в системі, якщо ви введете пароль цільового користувача. У випадку користувача root повідомлення пароля (багатьом) користувачам створює ризик для системи та є дуже поганою практикою безпеки. Основним використанням `su` є `su - target-username`. Однак, якщо змінити на root, цільове ім'я користувача є необов'язковим:

```
carol@debian:~$ su - root
Password:
root@debian:~# exit
logout
carol@debian:~$ su -
Password:
root@debian:~#
```

Використання тире (`-`) забезпечує завантаження цільового середовища користувача. Без нього старе середовище користувача буде збережено:

```
carol@debian:~$ su
Password:
root@debian:/home/carol#
```

З іншого боку, є команда `sudo`. За допомогою неї ви можете виконати команду як користувач root або будь-який інший користувач. З точки зору безпеки, sudo є набагато кращим варіантом, ніж su, оскільки вона має дві основні переваги:

1. щоб запустити команду від імені root, вам не потрібен пароль користувача root, а лише пароль користувача, який її викликає, відповідно до політики безпеки. Політикою

безпеки за замовчуванням є `sudoers`, як зазначено в `/etc/sudoers` і `/etc/sudoers.d/*`.

2. `sudo` дозволяє запускати окремі команди з підвищеними привілеями замість того, щоб запускати цілком нову підоболонку для root, як це робить `su`.

Основним використанням `sudo` є `sudo -u target-username command`. Однак, щоб запустити команду від імені користувача root, параметр `-u target-username` не потрібен:

```
carol@debian:~$ sudo -u mimi whoami
mimi
carol@debian:~$ sudo whoami
root
```

NOTE

`sudoers` використовуватиме мітку часу для кожного користувача (і кожного терміналу) для кешування облікових даних, так що ви можете використовувати `sudo` без пароля протягом п'ятнадцяти хвилин за умовчанням. Це значення за умовчанням можна змінити, додавши параметр `timestamp_timeout` як параметр `Defaults` в `/etc/sudoers` (наприклад, `Defaults timestamp_timeout=1` встановить час очікування кешування облікових даних на одну хвилину).

Файл `/etc/sudoers`

Головний конфігураційний файл `sudo`'s — це `/etc/sudoers` (існує також каталог `/etc/sudoers.d`). Це місце, де визначаються привілеї користувачів `sudo`. Іншими словами, тут ви вказуєте, хто може виконувати які команди, а саме які користувачі на яких машинах, а також інші налаштування. Використовується такий синтаксис:

```
carol@debian:~$ sudo less /etc/sudoers
(...)
# User privilege specification
root    ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
(...)
```

Специфікація привілеїв для користувача root така: `ALL=(ALL:ALL) ALL`. Це перекладається так: користувач root (`root`) може входити з усіх хостів (`ALL`), як усі користувачі та всі групи (`(ALL:ALL)`), і виконувати всі команди (`ALL`). Те саме стосується членів групи `sudo` — зверніть увагу, як назви груп ідентифікуються, починаючи зі знака відсотка (`%`).

Таким чином, щоб користувач `carol` міг перевіряти статус `apache2` з будь-якого хосту як будь-який користувач або група, ви додасте такий рядок у файл `sudoers`:

```
carol   ALL=(ALL:ALL) /usr/bin/systemctl status apache2
```

Можливо, ви захочете позбавити `carol` незручності від необхідності вводити свій пароль для запуску команди `systemctl status apache2`. Для цього ви зміните рядок, щоб він виглядав так:

```
carol   ALL=(ALL:ALL) NOPASSWD: /usr/bin/systemctl status apache2
```

Скажімо, тепер ви хочете обмежити свої хости 192.168.1.7 і ввімкнути `carol` для запуску `systemctl status apache2` від імені користувача `mimi`. Ви б змінили рядок таким чином:

```
carol   192.168.1.7=(mimi) /usr/bin/systemctl status apache2
```

Тепер ви можете перевірити статус веб-сервера Apache як користувач `mimi`:

```
carol@debian:~$ sudo -u mimi systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-06-09 13:12:19 CEST; 29min ago
(...)
```

Якщо `carol` має бути підвищено до системного адміністратора, і ви хочете надати їй усі привілеї, найпростішим підходом було б включити її до спеціальної групи `sudo` з `usermod` і опцією `-G` (ви також можете хочете використовувати опцію `-a`, яка гарантує, що користувача не буде видалено з інших груп, до яких він може належати):

```
root@debian:~# sudo useradd -aG sudo carol
```

NOTE У сімействі дистрибутивів Red Hat група `wheel` є аналогом спеціальної адміністративної групи `sudo` систем Debian.

Замість того, щоб безпосередньо редагувати `/etc/sudoers`, вам слід просто використати команду `visudo` від імені користувача (наприклад: `visudo`), яка відкриє `/etc/sudoers` за допомогою попередньо визначеного текстового редактора. Щоб змінити текстовий редактор за замовчуванням, ви можете додати опцію `editor` як налаштування `Defaults` в

`/etc/sudoers`. Наприклад, щоб змінити редактор на `nano`, потрібно додати такий рядок:

```
Defaults    editor=/usr/bin/nano
```

NOTE Крім того, ви можете вказати текстовий редактор за допомогою змінної середовища `EDITOR` під час використання `visudo` (наприклад: `EDITOR=/usr/bin/nano visudo`)

Окрім користувачів і груп, ви також можете використовувати псевдоніми в `/etc/sudoers`. Ви можете визначити три основні категорії псевдонімів: *псевдоніми хостів* (`Host_Alias`), *псевдоніми користувача* (`User_Alias`) і *псевдоніми команд* (`Cmnd_Alias`). Ось приклад:

```
# Host alias specification

Host_Alias SERVERS = 192.168.1.7, server1, server2

# User alias specification

User_Alias REGULAR_USERS = john, mary, alex

User_Alias PRIVILEGED_USERS = mimi

User_Alias ADMINS = carol, %sudo, PRIVILEGED_USERS, !REGULAR_USERS

# Cmnd alias specification

Cmnd_Alias SERVICES = /usr/bin/systemctl *

# User privilege specification
root    ALL=(ALL:ALL) ALL
ADMINS  SERVERS=SERVICES

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Розглядаючи цей зразок файлу `sudoers`, давайте пояснимо три типи псевдонімів трохи детальніше:

Псевдоніми хостів

включають список розділених комами імен хостів, IP-адрес, а також мереж і мережних груп (перед ними стоїть `+`). Також можна вказати мережні маски. Псевдонім хоста

`SERVERS` містить IP-адресу та два імені хостів:

```
Host_Alias SERVERS = 192.168.1.7, server1, server2
```

Псевдоніми користувачів

включають розділений комами список користувачів, визначений як імена користувачів, групи (починаються з `%`) і мережних груп (перед `+`). Ви можете виключити окремих користувачів за допомогою `!`. Псевдонім користувача `ADMINS`, наприклад, включає користувача `carol`, членів групи `sudo` і тих членів псевдоніма користувача `PRIVILEGE_USERS`, які не належать до псевдоніму користувача `REGULAR_USERS`:

```
User_Alias ADMINS = carol, %sudo, PRIVILEGED_USERS, !REGULAR_USERS
```

Псевдоніми команд

включають список команд і каталогів, розділених комами. Якщо вказано каталог, будь-який файл у цьому каталозі буде включено, але підкаталоги ігноруватимуться. Псевдонім команди `SERVICES` включає одну команду з усіма її підкомандами, як зазначено зірочкою (`*`):

```
Cmnd_Alias SERVICES = /usr/bin/systemctl *
```

У результаті специфікації псевдоніма рядок `ADMINS SERVERS=SERVICES` у розділі `User privilege specification` інтерпретується як: усі користувачі, що належать до `ADMINS`, можуть використовувати `sudo` для запуску будь-якої команди в `SERVICES` на будь-якому сервері в `SERVERS`.

NOTE

Існує четвертий тип псевдонімів, які можна включити в `/etc/sudoers`: *runas aliases* (`Runas_Alias`). Вони дуже схожі на псевдоніми користувачів, але дозволяють вказувати користувачів за їхнім *ідентифікатором користувача* (UID). Ця функція може бути зручною в деяких сценаріях.

Вправи до посібника

1. Заповніть наступну таблицю щодо спеціальних дозволів:

Спеціальний дозвіл	**Числове представлення**	**Символьне представлення**	**Знайти файли з *лише* цим дозволом**
`SUID`			
`SGID`			

2. Відображення файлів із *лише* встановленим бітом `SUID` або `SGID` зазвичай не дуже практично. Виконайте такі завдання, щоб довести, що ваші пошуки можуть бути продуктивнішими:
   - Знайдіть усі файли з `SUID` (та іншими дозволами), встановленими в `/usr/bin`:
   - Знайдіть усі файли з `SGID` (та іншими дозволами), встановленими в `/usr/bin`:
   - Знайдіть усі файли з `SUID` або `SGID`, встановленими в `/usr/bin`:

3. `chage` дозволяє змінити інформацію про закінчення терміну дії пароля користувача. Як root заповніть наступну таблицю, вказавши правильні команди для користувача `mary`:

Значення	**Команди** `chage`
Зробіть пароль дійсним протягом 365 днів.	
Примусьте користувача змінити пароль під час наступного входу.	
Установіть мінімальну кількість днів між змінами пароля на 1.	
Вимкніть термін дії пароля.	
Дозвольте користувачу змінити свій пароль у будь-який час.	
Встановіть період попередження на 7 днів і термін дії облікового запису до 20 серпня 2050 року.	

Значення	**Команди** `chage`
Роздрукуйте інформацію про термін дії поточного пароля користувача.	

4. Заповніть наступну таблицю відповідною мережевою утилітою:

Дія	**Команда(и)**
Показати мережні файли для хоста 192.168.1.55 на порту 22 за допомогою `lsof`.	
Показати процеси, які звертаються до стандартного порту веб-сервера Apache на вашій машині за допомогою `fuser`.	
Перерахувати всі *udp* сокети, які прослуховуються, на вашій машині за допомогою `netstat`.	
Просканувати порти від `80` до `443` на хості `192.168.1.55` за допомогою `nmap`.	

5. Виконайте такі завдання щодо *резидентного розміру* (*RSS*) і `ulimit` як звичайний користувач:
   - Відображення *м'яких* обмежень на *максимальний RSS*:
   - Показати *жорсткі* обмеження на *максимальний RSS*:
   - Встановіть *м'які* обмеження на *максимальний RSS* до 5000 кілобайт:
   - Встановіть *жорсткі* обмеження на *максимальний RSS* до 10 000 кілобайт:
   - Нарешті, спробуйте збільшити *жорстке* обмеження на *максимальний RSS* до 15 000 кілобайт. Ви можете зробити це? Чому?
6. Розгляньте наступний рядок виведення команди `last` і дайте відповіді на запитання:

```
carol    pts/0        192.168.1.4      Sun May 31 14:16 - 14:22  (00:06)
```

- Чи `carol` підключилася з віддаленого хосту? Чому?

- Як довго тривала сесія `carol`?

- Чи була `carol` підключена через справжній класичний текстовий термінал? Чому?

7. Розгляньте наступну частину файлу `/etc/sudoers` і дайте відповідь на запитання нижче.

```
# Host alias specification

Host_Alias SERVERS = 192.168.1.7, server1, server2

# User alias specification

User_Alias REGULAR_USERS = john, mary, alex

User_Alias PRIVILEGED_USERS = mimi

User_Alias ADMINS = carol, %sudo, PRIVILEGED_USERS, !REGULAR_USERS

# Cmnd alias specification

Cmnd_Alias WEB_SERVER_STATUS = /usr/bin/systemctl status apache2

# User privilege specification
root    ALL=(ALL:ALL) ALL
ADMINS  SERVERS=WEB_SERVER_STATUS

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Чи може `alex` перевірити статус веб-сервера Apache на будь-якому хості? Чому?

Дослідницькі вправи

1. Крім `SUID` і `SGID`, існує третій спеціальний дозвіл: *sticky bit*. Зараз він переважно використовується в таких каталогах, як `/tmp`, щоб запобігти звичайним користувачам видаляти або переміщувати файли, окрім своїх власних. Виконайте такі завдання:
   - Встановіть *sticky bit* на `~/temporal`:
   - Знайдіть каталоги з *sticky bit* (та будь-якими іншими дозволами), встановленими у вашому домашньому каталозі:
   - Зніміть *sticky bit* з `~/temporal`:
2. Коли пароль користувача заблоковано за допомогою `passwd -l username` або `usermod -L username`, як ви можете дізнатися про це, переглянувши файл `/etc/shadow`?
3. Яка команда `usermod` є аналогом `chage -E date username` або `chage --expiredate date username`?
4. Надайте дві різні команди `nmap` для сканування всіх 65535 портів на локальному хості:

Підсумки

У цьому уроці ви навчилися виконувати низку завдань адміністрування безпеки. Були розглянуті наступні теми:

- Пошук файлів зі спеціальним набором дозволів `SUID` і `SGID`.
- Встановлення та зміна паролів користувачів і робота з інформацією про вік паролів.
- Використання низки мережних утиліт для виявлення відкритих портів на хостах/мережах.
- Встановлення обмежень на системні ресурси.
- Перевірка користувачів, які входили до системи або які зараз увійшли в систему.
- Базове використання та конфігурація `sudo` (через файл `/etc/sudoers`).

Команди та файли, які розглянуто в цьому уроці:

`find`
: Шукати файли в ієрархії каталогів.

`passwd`
: Змінити пароль користувача.

`chmod`
: Змінити біти режиму файлу.

`chage`
: Змінити інформацію про термін дії пароля користувача.

`lsof`
: Вивести перелік відкритих файлів.

`fuser`
: Ідентифікація процесів за допомогою файлів або сокетів.

`netstat`
: Виведення мережних підключень.

`nmap`
: Інструмент дослідження мережі та сканер портів.

`ulimit`

Отримати та встановити ліміти користувачів.

`/etc/security/limits.conf`

Файл конфігурації для застосування обмежень для користувачів.

`last`

Вивести список останніх користувачів, які входили в систему.

`lastb`

Вивести список невдалих спроб входу.

`/var/log/wtmp`

База даних логінів користувачів.

`who`

Показати, хто ввійшов.

`w`

Показати, хто ввійшов і що вони роблять.

`su`

Змінити користувача або стати суперкористувачем.

`sudo`

Виконати команду від імені іншого користувача (включаючи суперкористувача).

`/etc/sudoers`

Файл конфігурації за умовчанням для політики безпеки `sudo`.

Відповіді до вправ посібника

1. Заповніть наступну таблицю щодо спеціальних дозволів:

Спеціальний дозвіл	**Числове представлення**	**Символьне представлення**	**Знайти файли з *лише* цим дозволом**
`SUID`	`4000`	`s,S`	`find -perm 4000`, `find -perm u+s`
`SGID`	`2000`	`s,S`	`find -perm 2000`, `find -perm g+s`

2. Відображення файлів із *лише* встановленим бітом `SUID` або `SGID` зазвичай не дуже практично. Виконайте такі завдання, щоб довести, що ваші пошуки можуть бути продуктивнішими:
   - Знайдіть усі файли з `SUID` (та іншими дозволами), встановленими в `/usr/bin`:

     `find /usr/bin -perm -4000` або `find /usr/bin -perm -u+s`
   - Знайдіть усі файли з `SGID` (та іншими дозволами), встановленими в `/usr/bin`:

     `find /usr/bin -perm -2000` або `find /usr/bin -perm -g+s`
   - Знайдіть усі файли з `SUID` або `SGID`, встановленими в `/usr/bin`:

     `find /usr/bin -perm /6000`

3. `chage` дозволяє змінити інформацію про закінчення терміну дії пароля користувача. Як root заповніть наступну таблицю, вказавши правильні команди для користувача `mary`:

Значення	**Команди** `chage`
Зробіть пароль дійсним протягом 365 днів.	`chage -M 365 mary`, `chage --maxdays 365 mary`
Примусьте користувача змінити пароль під час наступного входу.	`chage -d 0 mary`, `chage --lastday 0 mary`
Установіть мінімальну кількість днів між змінами пароля на 1.	`chage -m 1 mary`, `chage --mindays 1 mary`
Вимкніть термін дії пароля.	`chage -M 99999 mary`, `chage --maxdays 99999 mary`

Значення	**Команди** `chage`
Дозвольте користувачу змінити свій пароль у будь-який час.	`chage -m 0 mary`, `chage --mindays 0 mary`
Встановіть період попередження на 7 днів і термін дії облікового запису до 20 серпня 2050 року.	`chage -W 7 -E 2050-08-20 mary`, `chage --warndays 7 --expiredate 2050-08-20 mary`
Роздрукуйте інформацію про термін дії поточного пароля користувача.	`chage -l mary`, `chage --list mary`

4. Заповніть наступну таблицю відповідною мережевою утилітою:

Дія	**Команда(и)**
Показати мережні файли для хоста 192.168.1.55 на порту 22 за допомогою `lsof`.	`lsof -i@192.168.1.55:22`
Показати процеси, які звертаються до стандартного порту веб-сервера Apache на вашій машині за допомогою `fuser`.	`fuser -vn tcp 80`, `fuser --verbose --namespace tcp 80`
Перерахувати всі *udp* сокети, які прослуховуються, на вашій машині за допомогою `netstat`.	`netstat -lu`,`netstat --listening --udp`
Просканувати порти від `80` до `443` на хості `192.168.1.55` за допомогою `nmap`.	`nmap -p 80-443 192.168.1.55`

5. Виконайте такі завдання щодо *резидентного розміру* (*RSS*) і `ulimit` як звичайний користувач:
   - Відображення *м'яких* обмежень на *максимальний RSS*:

     `ulimit -m`, `ulimit -Sm`
   - Показати *жорсткі* обмеження на *максимальний RSS*:

     `ulimit -Hm`
   - Встановіть *м'які* обмеження на *максимальний RSS* до 5000 кілобайт:

     `ulimit -Sm 5000`
   - Встановіть *жорсткі* обмеження на *максимальний RSS* до 10 000 кілобайт:

`ulimit -Hm 10000`

- Нарешті, спробуйте збільшити *жорстке* обмеження на *максимальний RSS* до 15 000 кілобайт. Ви можете зробити це? Чому?

  Ні. Після встановлення звичайні користувачі не можуть збільшити жорсткі обмеження.

6. Розгляньте наступний рядок виведення команди `last` і дайте відповіді на запитання:

```
carol    pts/0        192.168.1.4      Sun May 31 14:16 - 14:22  (00:06)
```

- Чи `carol` підключилася з віддаленого хосту? Чому?

  Так, IP-адреса віддаленого хоста вказана в третьому стовпці.

- Як довго тривала сесія `carol`?

  Шість хвилин (як показано в останньому стовпчику).

- Чи була `carol` підключена через справжній класичний текстовий термінал? Чому?

  Ні, `pts/0` у другому стовпці вказує на те, що підключення було здійснено через графічний емулятор терміналу (він же *Pseudo Terminal Slave*).

7. Розгляньте наступну частину файлу `/etc/sudoers` і дайте відповідь на запитання нижче.

```
# Host alias specification

Host_Alias SERVERS = 192.168.1.7, server1, server2

# User alias specification

User_Alias REGULAR_USERS = john, mary, alex

User_Alias PRIVILEGED_USERS = mimi

User_Alias ADMINS = carol, %sudo, PRIVILEGED_USERS, !REGULAR_USERS

# Cmnd alias specification

Cmnd_Alias WEB_SERVER_STATUS = /usr/bin/systemctl status apache2
```

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
ADMINS  SERVERS=WEB_SERVER_STATUS

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Чи може `alex` перевірити статус веб-сервера Apache на будь-якому хості? Чому?

Ні, оскільки він є членом `REGULAR_USERS`, а ця група користувачів виключена з `ADMINS`; єдині користувачі (крім `carol`, членів групи `sudo` і `root`), які можуть запускати `systemctl status apache2` на `SERVERS`.

Відповіді до дослідницьких вправ

1. Крім `SUID` і `SGID`, існує третій спеціальний дозвіл: *sticky bit*. Зараз він переважно використовується в таких каталогах, як `/tmp`, щоб запобігти звичайним користувачам видаляти або переміщувати файли, окрім своїх власних. Виконайте такі завдання:
   - Встановіть *sticky bit* на `~/temporal`:

     `chmod +t temporal`, `chmod 1755 temporal`

   - Знайдіть каталоги з *sticky bit* (та будь-якими іншими дозволами), встановленими у вашому домашньому каталозі:

     `find ~ -perm -1000`, `find ~ -perm /1000`

   - Зніміть *sticky bit* з `~/temporal`:

     `chmod -t temporal`, `chmod 0755 temporal`

2. Коли пароль користувача заблоковано за допомогою `passwd -l username` або `usermod -L username`, як ви можете дізнатися про це, переглянувши файл `/etc/shadow`?

   Знак оклику з'явиться у другому полі одразу після імені користувача, якого це стосується (наприклад, `mary:!$6$gOg9xJgv...`).

3. Яка команда `usermod` є аналогом `chage -E date username` або `chage --expiredate date username`?

   `usermod -e date username`, `usermod --expiredate date username`

4. Надайте дві різні команди `nmap` для сканування всіх 65535 портів на локальному хості:

   `nmap -p 1-65535 localhost` та `nmap -p- localhost`

110.2 Налаштування безпеки хостау

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 110.2

Ваговий коефіцієнт

3

Ключові галузі знань

- Знання тіньових паролів і їх роботи.
- Вимкнення мережевих служб, які не використовуються.
- Розуміння ролі TCP-обгорток.

Частковий список файлів, термінів та утиліт, що використовуються

- `/etc/nologin`
- `/etc/passwd`
- `/etc/shadow`
- `/etc/xinetd.d/`
- `/etc/xinetd.conf`
- `systemd.socket`
- `/etc/inittab`
- `/etc/init.d/`
- `/etc/hosts.allow`
- `/etc/hosts.deny`

Linux Professional Institute

110.2 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	110 Безпека
Тема:	110.2 Налаштування безпеки хоста
Урок:	1 з 1

Вступ

У цьому розділі пояснюється чотири основні способи підвищення безпеки хоста:

1. Деякі основні команди та параметри конфігурації для покращення безпеки автентифікації за допомогою тіньових паролів.
2. Як використовувати супердемони для прослуховування вхідних мережних з'єднань.
3. Перевірка мережних служб на непотрібні демони.
4. Обгортки TCP у якості простого брандмауера.

Покращення безпеки автентифікації за допомогою тіньових паролів

Основні компоненти даних облікового запису користувача зберігаються у файлі `/etc/passwd`. Цей файл містить сім полів: ім'я для входу, ідентифікатор користувача, ідентифікатор групи, пароль, коментар (він же GECOS), розташування домашнього каталогу та, нарешті, оболонку за замовчуванням. Простий спосіб запам'ятати порядок

цих полів — подумати про процес входу користувача: спочатку ви вводите ім'я для входу, по-друге, система зіставить його з ідентифікатором користувача (uid), а по-третє, ідентифікатором групи (gid). Четвертий крок запитує пароль, п'ятий шукає коментар, шостий визначає домашній каталог користувача, а сьомий крок встановлює оболонку за замовчуванням.

Хоча в сучасних системах пароль більше не зберігається у файлі `/etc/passwd`. Натомість поле пароля містить лише літеру `x` у нижньому регістрі. Файл `/etc/passwd` має бути доступним для читання всім користувачам. Тому не варто зберігати там паролі. `x` означає, що зашифрований (хешований) пароль фактично зберігається у файлі `/etc/shadow`. Цей файл не повинен бути доступним для читання всім користувачам.

Параметри пароля налаштовуються за допомогою команд `passwd` і `chage`. Обидві команди змінять запис для користувача `emma` у файлі `/etc/shadow`. Як суперкористувач ви можете встановити пароль для користувача `emma` за допомогою такої команди:

```
$ sudo passwd emma
New password:
Retype new password:
passwd: password updated successfully
```

Потім вам двічі буде запропоновано підтвердити новий пароль.

Щоб переглянути час закінчення терміну дії пароля та інші налаштування терміну дії пароля для користувача `emma`, виконайте наступні дії:

```
$ sudo chage -l emma
Last password change                    : Apr 27, 2020
Password expires                    : never
Password inactive                   : never
Account expires                     : never
Minimum number of days between password change      : 0
Maximum number of days between password change      : 99999
Number of days of warning before password expires   : 7
```

Щоб запобігти входу користувача `emma` в систему, суперкористувач може встановити термін дії пароля, який передує поточній даті. Наприклад, якщо сьогоднішня дата 2020-03-27, ви можете закінчити термін дії пароля користувача, використовуючи старішу дату:

```
$ sudo chage -E 2020-03-26 emma
```

Крім того, суперкористувач може використовувати наступний варіант:

```
$ sudo passwd -l emma
```

щоб тимчасово заблокувати обліковий запис за допомогою опції `-l` для `passwd`. Щоб перевірити вплив цих змін, спробуйте увійти з обліковим записом `emma`:

```
$ sudo login emma
Password:
Your account has expired; please contact your system administrator

Authentication failure
```

Щоб запобігти тимчасово входу в систему всіх користувачів, окрім користувача root, суперкористувач може створити файл під назвою `/etc/nologin`. Цей файл може містити повідомлення для користувачів про те, чому вони не можуть увійти (наприклад, сповіщення про технічне обслуговування системи). Для отримання додаткової інформації див. `man 5 nologin`. Зауважте, що існує також команда `nologin`, яку можна використати для запобігання входу, якщо її встановлено як оболонку за замовчуванням для користувача. Наприклад:

```
$ sudo usermod -s /sbin/nologin emma
```

Дивіться `man 8 nologin` для отримання додаткової інформації.

Як використовувати Superdaemon для прослуховування вхідних мережних підключень

Мережні служби, такі як веб-сервери, сервери електронної пошти та сервери друку, зазвичай працюють як окрема служба, яка прослуховує виділений мережний порт. Усі ці автономні служби працюють паралельно. У класичній системі на основі Sys-V-init кожною з цих служб можна керувати командою `service`. У поточних системах на базі systemd використовується `systemctl` для керування службою.

У минулі часи доступність комп'ютерних ресурсів була набагато меншою. Запуск багатьох служб в автономному режимі в тандемі не був прийнятним варіантом. Замість цього використовувався так званий супердемон для прослуховування вхідних мережних з'єднань і запуску відповідної служби на вимогу. Цей спосіб побудови мережного підключення займав трохи більше часу. Добре відомими супердемонами є `inetd` і `xinetd`.

У поточних системах, заснованих на systemd, блок `systemd.socket` можна використовувати подібним чином. У цьому розділі ми будемо використовувати `xinetd` для перехоплення з'єднань із демоном `sshd` і запускати цей демон за запитом, щоб продемонструвати, як використовувався супердемон.

Перед налаштуванням служби `xinetd` необхідна певна підготовка. Немає значення, чи використовуєте ви систему на основі Debian чи Red Hat. Хоча ці пояснення перевірено на Debian/GNU Linux 9.9, вони мають працювати на будь-якій поточній системі Linux із системою systemd без будь-яких суттєвих змін. Спочатку переконайтеся, що встановлено пакунки `openssh-server` і `xinetd`. Тепер переконайтеся, що служба SSH працює з:

```
$ systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-04-27 09:33:48 EDT; 3h 11min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 430 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 460 (sshd)
    Tasks: 1 (limit: 1119)
   Memory: 5.3M
   CGroup: /system.slice/ssh.service
           └─460 /usr/sbin/sshd -D
```

Також перевірте, чи служба SSH прослуховує стандартний мережний порт 22:

```
# lsof -i :22
COMMAND  PID USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
sshd    1194 root    3u  IPv4 16053268      0t0  TCP *:ssh (LISTEN)
sshd    1194 root    4u  IPv6 16053270      0t0  TCP *:ssh (LISTEN)
```

Нарешті зупиніть службу SSH за допомогою:

```
$ sudo systemctl stop sshd.service
```

У випадку, якщо ви бажаєте зробити цю зміну постійною і після перезавантаження, використовуйте `systemctl disable sshd.service`.

Тепер ви можете створити файл конфігурації xinetd `/etc/xinetd.d/ssh` з деякими основними налаштуваннями:

```
service ssh
{
    disable     = no
    socket_type = stream
    protocol    = tcp
    wait        = no
    user        = root
    server      = /usr/sbin/sshd
    server_args     = -i
    flags       = IPv4
    interface   = 192.168.178.1
}
```

Перезапустіть службу xinetd за допомогою:

```
$ sudo systemctl restart xinetd.service
```

Перевірте, яка служба зараз прослуховує вхідні з'єднання SSH.

```
$ sudo lsof -i :22
COMMAND   PID USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
xinetd  24098 root    5u  IPv4 7345141      0t0  TCP 192.168.178.1:ssh (LISTEN)
```

Ми бачимо, що служба xinetd взяла під контроль доступ до порту 22.

Ось деякі додаткові відомості про конфігурацію `xinetd`. Основний файл конфігурації – `/etc/xinetd.conf`:

```
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/

defaults
{

# Please note that you need a log_type line to be able to use log_on_success
# and log_on_failure. The default is the following :
# log_type = SYSLOG daemon info

}
```

```
includedir /etc/xinetd.d
```

Окрім налаштувань за замовчуванням, існує лише одна директива для встановлення каталогу включення. У цьому каталозі ви можете налаштувати окремий файл конфігурації для кожної служби, яку ви хочете обробляти `xinetd`. Ми зробили це вище для служби SSH і назвали файл `/etc/xinetd.d/ssh`. Назви файлів конфігурації можна вибирати довільно, за винятком імен файлів, які містять крапку (`.`) або закінчуються тильдою (`~`). Але поширеною практикою є назвати файл за назвою служби, яку ви хочете налаштувати.

Деякі конфігураційні файли в каталозі `/etc/xinet.d/` вже надаються дистрибутивом:

```
$ ls -l /etc/xinetd.d
total 52
-rw-r--r-- 1 root root 640 Feb  5  2018 chargen
-rw-r--r-- 1 root root 313 Feb  5  2018 chargen-udp
-rw-r--r-- 1 root root 502 Apr 11 10:18 daytime
-rw-r--r-- 1 root root 313 Feb  5  2018 daytime-udp
-rw-r--r-- 1 root root 391 Feb  5  2018 discard
-rw-r--r-- 1 root root 312 Feb  5  2018 discard-udp
-rw-r--r-- 1 root root 422 Feb  5  2018 echo
-rw-r--r-- 1 root root 304 Feb  5  2018 echo-udp
-rw-r--r-- 1 root root 312 Feb  5  2018 servers
-rw-r--r-- 1 root root 314 Feb  5  2018 services
-rw-r--r-- 1 root root 569 Feb  5  2018 time
-rw-r--r-- 1 root root 313 Feb  5  2018 time-udp
```

Ці файли можна використовувати як шаблони у рідкісних випадках, коли вам доведеться використовувати деякі застарілі служби, такі як `daytime`, дуже рання реалізація сервера часу. Усі ці файли шаблонів містять директиву `disable = yes`.

Ось деякі додаткові відомості про директиви, які використовуються у файлі прикладу `/etc/xinetd.d/ssh` для ssh вище.

```
service ssh
{
    disable     = no
    socket_type = stream
    protocol    = tcp
    wait        = no
    user        = root
    server      = /usr/sbin/sshd
    server_args     = -i
```

```
    flags       = IPv4
    interface   = 192.168.178.1
}
```

`service`

Виводить службу, якою має керувати xinetd. Ви можете використовувати або номер порту, наприклад 22, або назву, зіставлену з номером порту в `/etc/services`, наприклад. `ssh`.

`{`

Детальні параметри починаються з відкриваючої фігурної дужки.

`disable`

Щоб активувати ці параметри, встановіть значення `no`. Якщо ви хочете тимчасово вимкнути це налаштування, ви можете встановити для нього значення `yes`.

`socket_type`

Ви можете вибрати `stream` для сокетів TCP або `dgram` для сокетів UDP тощо.

`protocol`

Виберіть TCP або UDP.

`wait`

Для TCP-з'єднань зазвичай встановлено значення `no`.

`user`

Сервіс, запущений у цьому рядку, буде належати цьому користувачеві.

`server`

Повний шлях до служби, яку має запустити `xinetd`.

`server_args`

Тут можна додати опції сдужби. Якщо запущено суперсервером, для багатьох служб потрібна спеціальна опція. Для SSH це буде параметр `-i`.

`flags`

Ви можете вибрати IPv4, IPv6 та інші.

`interface`

Мережний інтерфейс, яким має керувати `xinetd`. Примітка: ви також можете вибрати

директиву `bind`, яка є лише синонімом `interface`.

`}`

Завершіть закриттям фігурної дужки.

Спадкоємцями служб, запущених суперсервером `xinetd`, є модулі сокетів systemd. Налаштування модуля сокета systemd дуже просте, тому що вже є попередньо визначений модуль сокета systemd для SSH. Переконайтеся, що служби `xinetd` і SSH не запущені.

Тепер вам просто потрібно запустити модуль SSH-сокета:

```
$ sudo systemctl start ssh.socket
```

Щоб перевірити, яка служба зараз прослуховує порт 22, ми знову використовуємо `lsof`. Зауважте, що опція `-P` була використана для показу номера порту замість назви служби у виведенні:

```
$ sudo lsof -i :22 -P
COMMAND PID USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
systemd   1 root   57u  IPv6 14730112      0t0  TCP *:22 (LISTEN)
```

Щоб завершити цей сеанс, ви повинні спробувати ввійти на свій сервер за допомогою свого обраного SSH-клієнта.

TIP Якщо `systemctl start ssh.socket` не працює з вашим дистрибутивом, спробуйте `systemctl start sshd.socket`.

Перевірка служб на непотрібні демони

З міркувань безпеки, а також для контролю системних ресурсів важливо знати, які служби працюють. Служби, які непотрібні та не використовуються, слід відключити. Наприклад, якщо вам не потрібно розповсюджувати веб-сторінки, не потрібно запускати веб-сервер, такий як Apache або nginx.

У системах на основі SyS-V-init ви можете перевірити стан усіх служб за допомогою наступної команди:

```
$ sudo service --status-all
```

Перевірте, чи потрібні всі служби, перелічені у виведенні команди, і вимкніть усі

непотрібні служби для систем на основі Debian:

```
$ sudo update-rc.d SERVICE-NAME remove
```

або в системах на основі Red Hat:

```
$ sudo chkconfig SERVICE-NAME off
```

У сучасних системах на базі systemd ми можемо використовувати наступне, щоб вивести всі запущені служби:

```
$ systemctl list-units --state active --type service
```

Потім ви вимкнете кожну непотрібну службу за допомогою:

```
$ sudo systemctl disable UNIT --now
```

Ця команда зупинить службу та видалить її зі списку служб, щоб запобігти її запуску під час наступного завантаження системи.

Крім того, щоб отримати перелік мережних служб, які перебувають в стані прослуховування, ви можете використовувати `netstat` на старих системах (за умови, що у вас встановлено пакет `net-tools`):

```
$ netstat -ltu
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 localhost:mysql         0.0.0.0:*               LISTEN
tcp6       0      0 [::]:http               [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
```

Або в сучасних системах ви можете використати еквівалентну команду `ss` (для “служб сокетів”):

```
$ ss -ltu
Netid      State         Recv-Q        Send-Q      Local Address:Port           Peer
```

```
Address:Port
udp        UNCONN       0            0                0.0.0.0:bootpc
0.0.0.0:*
tcp        LISTEN       0            128              0.0.0.0:ssh
0.0.0.0:*
tcp        LISTEN       0            80             127.0.0.1:mysql
0.0.0.0:*
tcp        LISTEN       0            128                 *:http
*:*
tcp        LISTEN       0            128                [::]:ssh
[::]:*
```

TCP обгортки як простий міжмережний екран

У часи, коли для Linux не було доступних брандмауерів, для захисту мережних з'єднань із хостом використовувалися TCP-обгортки. Зараз багато програм більше не працюють під обгортками TCP. В останніх дистрибутивах Red Hat (наприклад, Fedora 29) підтримку TCP-обгорток було повністю вилучено. Щоб підтримувати застарілі системи Linux, які все ще використовують TCP-обгортки, корисно мати деякі базові знання про цю конкретну технологію.

Ми ще раз використаємо SSH-службу як базовий приклад. Сервіс в нашому прикладі хосту має бути доступним лише з локальної мережі. Спочатку ми перевіряємо, чи використовує SSH-демон бібліотеку `libwrap`, яка пропонує підтримку TCP-обгорток:

```
$ ldd /usr/sbin/sshd | grep "libwrap"
        libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0 (0x00007f91dbec0000)
```

Тепер ми додаємо такий рядок у файл `/etc/hosts.deny`:

```
sshd: ALL
```

Нарешті ми налаштовуємо виняток у файлі `/etc/hosts.allow` для SSH-з'єднань з локальної мережі:

```
sshd: LOCAL
```

Зміни набувають чинності зразу, перезапускати жодну службу не потрібно. Ви можете перевірити це за допомогою клієнта `ssh`.

Вправи до посібника

1. Як можна розблокувати раніше заблокований обліковий запис `emma`?

2. Раніше обліковий запис `emma` мав встановлений термін дії. Як можна встановити термін придатності на `never`?

3. Уявіть, що служба друку CUPS, яка обробляє завдання друку, не потрібна на вашому сервері. Як відключити послугу назавжди? Як перевірити, що відповідний порт більше не активний?

4. Ви встановили веб-сервер nginx. Як можна перевірити, чи підтримує nginx TCP-обгортки?

Дослідницькі вправи

1. Перевірте, чи наявність файлу `/etc/nologin` не перешкоджає входу користувача `root`?

2. Чи наявність файлу `/etc/nologin` запобігає входу без пароля за допомогою SSH-ключів?

3. Що відбувається під час входу, якщо файл `/etc/nologin` містить тільки цей рядок тексту `login currently is not possible`?

4. Чи може звичайний користувач `emma` отримати інформацію про користувача `root`, що міститься у файлі `/etc/passwd`, наприклад, за допомогою команди `grep root /etc/passwd`?

5. Чи може звичайний користувач `emma` отримати інформацію про свій власний хешований пароль, що міститься у файлі `/etc/shadow`, наприклад, за допомогою команди `grep emma /etc/shadow`?

6. Які кроки потрібно зробити, щоб увімкнути та перевірити застарілу службу daytime, яку оброблятиме xinetd? Зауважте, що це лише дослідницька вправа, не робіть цього у робочому середовищі.

Підсумки

На цьому уроці ви дізналися про:

1. У якому файлі зберігаються паролі, а також деякі налаштування безпеки паролів, наприклад, термін придатності.
2. Призначення супердемона `xinetd` і як його запустити, а також запуск служби `sshd` за вимогою.
3. Як перевірити, які мережні служби запущені та як вимкнути непотрібні служби.
4. Використання TCP-обгорток у якості простого міжмережного екрану.

Команди, які використовуються в лабораторних роботах і вправах:

`chage`

Змінити термін дії пароля користувача.

`chkconfig`

Класична команда, спочатку використовувалася в системах на базі Red Hat, щоб установити, чи запускатиметься служба під час завантаження чи ні.

`netstat`

Класична утиліта (тепер у пакеті `net-tools`), що відображає демони, які отримують доступ до мережних портів у системі та їхнє використання.

`nologin`

Команда, яку можна використовувати замість оболонки користувача, щоб запобігти входу в систему.

`passwd`

Використовується для створення або зміни пароля користувача.

`service`

Застарілий метод керування статусом демона, наприклад зупинка або запуск служби.

`ss`

Сучасний еквівалент `netstat`, але також відображає більше інформації про різні сокети, що використовуються в системі.

`systemctl`

Команда керування системою, яка використовується для керування різними аспектами служб і сокетів на комп'ютері за допомогою systemd.

`update-rc.d`

Класична команда, подібна до `chkconfig`, яка дозволяє або вимикає запуск системи під час завантаження дистрибутивів на основі Debian.

`xinetd`

Супердемон, який може контролювати доступ до мережної служби на вимогу, таким чином залишаючи службу неактивною, доки її фактично не викличуть для виконання певного завдання.

Відповіді до вправ посібника

1. Як можна розблокувати раніше заблокований обліковий запис `emma`?

   Суперкористувач може запустити `passwd -u emma`, щоб розблокувати обліковий запис.

2. Раніше обліковий запис `emma` мав встановлений термін дії. Як можна встановити термін придатності на `never`?

   Суперкористувач може використвати `chage -E -1 emma`, щоб встановити нескінченим термін дії. Цей параметр можна перевірити за допомогою `chage -l emma`.

3. Уявіть, що служба друку CUPS, яка обробляє завдання друку, не потрібна на вашому сервері. Як відключити послугу назавжди? Як перевірити, що відповідний порт більше не активний?

   Від імені суперкористувача запустити

   ```
   systemctl disable cups.service --now
   ```

   Тепер можна перевірити

   ```
   netstat -l | grep ":ipp "` or `ss -l | grep ":ipp "
   ```

4. Ви встановили веб-сервер nginx. Як можна перевірити, чи підтримує nginx TCP-обгортки?

   Команда

   ```
   ldd /usr/sbin/nginx | grep "libwrap"
   ```

   покаже запис, якщо nginx підтримує TCP-обгортки.

Відповіді до дослідницьких вправ

1. Перевірте, чи наявність файлу `/etc/nologin` не перешкоджає входу користувача `root`?

   Користувач `root` все ще може ввійти.

2. Чи наявність файлу `/etc/nologin` запобігає входу без пароля за допомогою SSH-ключів?

   Так, вхід без пароля також заборонено.

3. Що відбувається під час входу, якщо файл `/etc/nologin` містить тільки цей рядок тексту `login currently is not possible`?

   Буде показано повідомлення `login currently is not possible`, і вхід буде заборонено.

4. Чи може звичайний користувач `emma` отримати інформацію про користувача `root`, що міститься у файлі `/etc/passwd`, наприклад, за допомогою команди `grep root /etc/passwd`?

   Так, оскільки всі користувачі мають дозвіл на читання цього файлу.

5. Чи може звичайний користувач `emma` отримати інформацію про свій власний хешований пароль, що міститься у файлі `/etc/shadow`, наприклад, за допомогою команди `grep emma /etc/shadow`?

   Ні, оскільки звичайні користувачі не мають дозволу на читання цього файлу.

6. Які кроки потрібно зробити, щоб увімкнути та перевірити застарілу службу daytime, яку оброблятиме xinetd? Зауважте, що це лише дослідницька вправа, не робіть цього у робочому середовищі.

   Спочатку змініть файл `/etc/xinetd.d/daytime` і встановіть директиву `disable = no`. Потім перезапустіть службу xinetd `systemctl restart xinetd.service` (або `service xinetd restart` на системах із SyS-V-Init). Тепер ви можете перевірити, чи працює `nc localhost daytime`. Замість `nc` ви також можете використовувати `netcat`.

110.3 Захист даних за допомогою шифрування

Посилання на вимоги LPI

LPIC-1 version 5.0, Exam 102, Objective 110.3

Ваговий коефіцієнт

4

Ключові галузі знань

- Виконання базової конфігурації та використання клієнта OpenSSH 2.
- Розуміння ролі ключів хосту сервера OpenSSH 2.
- Виконання базової конфігурації, використання та відкликання GnuPG.
- Використання GPG для шифрування, дешифрування, підпису та перевірки файлів.
- Розуміння тунелів портів SSH (включаючи тунелі X11).

Частковий список файлів, термінів та утиліт, що використовуються

- `ssh`
- `ssh-keygen`
- `ssh-agent`
- `ssh-add`
- `~/.ssh/id_rsa` and `id_rsa.pub`
- `~/.ssh/id_dsa` and `id_dsa.pub`
- `~/.ssh/id_ecdsa` and `id_ecdsa.pub`
- `~/.ssh/id_ed25519` and `id_ed25519.pub`
- `/etc/ssh/ssh_host_rsa_key` and `ssh_host_rsa_key.pub`

- `/etc/ssh/ssh_host_dsa_key` and `ssh_host_dsa_key.pub`
- `/etc/ssh/ssh_host_ecdsa_key` and `ssh_host_ecdsa_key.pub`
- `/etc/ssh/ssh_host_ed25519_key` and `ssh_host_ed25519_key.pub`
- `~/.ssh/authorized_keys`
- `ssh_known_hosts`
- `gpg`
- `gpg-agent`
- `~/.gnupg/`

110.3 Урок 1

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	110 Безпека
Тема:	110.3 Захист даних за допомогою шифрування
Урок:	1 з 2

Вступ

Захист даних за допомогою шифрування має першочергове значення в багатьох аспектах сучасного системного адміністрування, особливо коли йдеться про віддалений доступ до систем. На відміну від небезпечних рішень, таких як *telnet*, *rlogin* або *FTP*, протокол *SSH* (*Secure Shell*) розроблено з урахуванням безпеки. Використовуючи криптографію з відкритим ключем, він автентифікує хости та користувачів і шифрує весь подальший обмін інформацією. Крім того, SSH можна використовувати для встановлення *портових тунелів*, які, серед іншого, дозволяють незашифрованому протоколу передавати дані через зашифроване SSH-з'єднання. Поточна рекомендована версія протоколу SSH – 2.0. *OpenSSH* — це безкоштовна реалізація SSH-протоколу з відкритим кодом.

Цей урок охопить основну конфігурацію клієнта *OpenSSH*, а також роль ключів хоста *OpenSSH* сервера. Також буде обговорено концепцію тунелів SSH-портів. Ми будемо використовувати дві машини з такими налаштуваннями:

Роль машини	ОС	IP-адреса	Ім’я хоста	Користувач
Клієнт	Debian GNU/Linux 10 (buster)	`192.168.1.55`	`debian`	`carol`
Сервер	openSUSE Leap 15.1	`192.168.1.77`	`halof`	`ina`

Базова конфігурація та використання клієнта OpenSSH

Хоча сервер OpenSSH і клієнт постачаються в окремих пакунках, зазвичай можна встановити метапакунок, який буде містити обидва одночасно. Щоб встановити віддалений сеанс із сервером SSH, ви використовуєте команду `ssh`, вказуючи користувача, якого ви хочете підключити на віддаленій машині, і IP-адресу віддаленої машини або ім’я хоста. Під час першого підключення до віддаленого хосту ви отримаєте таке повідомлення:

```
carol@debian:~$ ssh ina@192.168.1.77
The authenticity of host '192.168.1.77 (192.168.1.77)' can't be established.
ECDSA key fingerprint is SHA256:5JF7anupYipByCQm2BPvDHRVFJJixeslmppi2NwATYI.
Are you sure you want to continue connecting (yes/no)?
```

Після введення `yes` і натискання Enter вам буде запропоновано ввести пароль віддаленого користувача. У разі успішного введення ви побачите попередження, а потім увійдете на віддалений хост:

```
Warning: Permanently added '192.168.1.77' (ECDSA) to the list of known hosts.
Password:
Last login: Sat Jun 20 10:52:45 2020 from 192.168.1.4
Have a lot of fun...
ina@halof:~>
```

Повідомлення досить зрозумілі: оскільки ви вперше встановили з’єднання з віддаленим сервером `192.168.1.77`, його автентичність не можна було перевірити в жодній базі даних. Таким чином, віддалений сервер надав `ECDSA key fingerprint` свого відкритого ключа (за допомогою хеш-функції `SHA256`). Після того, як ви прийняли з’єднання, відкритий ключ віддаленого сервера було додано до бази даних *відомих хостів*, таким чином дозволяючи автентифікацію сервера для майбутніх з’єднань. Цей список відкритих ключів *відомих хостів* зберігається у файлі `known_hosts`, який знаходиться в `~/.ssh`:

```
ina@halof:~> exit
logout
Connection to 192.168.1.77 closed.
carol@debian:~$ ls .ssh/
known_hosts
```

`.ssh` і `known_hosts` були створені після встановлення першого віддаленого підключення. `~/.ssh` є каталогом за замовчуванням для конфігурації користувача та інформації автентифікації.

> **NOTE** Ви також можете використовувати `ssh`, щоб просто виконати одну команду на віддаленому хості, а потім повернутися до локального терміналу (наприклад, запустивши `ssh ina@halof ls`).

Якщо ви використовуєте одного користувача як на локальному, так і на віддаленому хостах, немає необхідності вказувати ім'я користувача під час встановлення з'єднання SSH. Наприклад, якщо ви ввійшли як користувач `carol` на `debian` і хочете підключитися до `halof` також як користувач `carol`, вам слід просто ввести `ssh 192.168.1.77` або `ssh halof` (якщо ім'я можна розпізнати):

```
carol@debian:~$ ssh halof
Password:
Last login: Wed Jul  1 23:45:02 2020 from 192.168.1.55
Have a lot of fun...
carol@halof:~>
```

Тепер припустимо, що ви встановлюєте нове віддалене з'єднання з хостом, який має ту саму IP-адресу, що й `halof` (звичайна річ, якщо ви використовуєте DHCP у своїй локальній мережі). Ви отримаєте попередження про можливість атаки *людина посередині*:

```
carol@debian:~$ ssh john@192.168.1.77
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:KH4q3vP6C7e0SEjyG8Wlz9fVlf+jmWJ5139RBxBh3TY.
Please contact your system administrator.
Add correct host key in /home/carol/.ssh/known_hosts to get rid of this message.
```

```
Offending ECDSA key in /home/carol/.ssh/known_hosts:1
  remove with:
  ssh-keygen -f "/home/carol/.ssh/known_hosts" -R "192.168.1.77"
ECDSA host key for 192.168.1.77 has changed and you have requested strict checking.
Host key verification failed.
```

Оскільки ви не маєте справу з атакою *людина посередині*, ви можете безпечно додати відбиток відкритого ключа (public key fingerprint) нового хоста до `.ssh/known_hosts`. Як зазначено в повідомленні, ви можете спочатку використати команду `ssh-keygen -f "/home/carol/.ssh/known_hosts" -R "192.168.1.77"`, щоб видалити *небезпечний* ключ (альтернативно, ви можете скористатися `ssh-keygen -R 192.168.1.77`, щоб видалити всі ключі, що належать до `192.168.1.77`) з `~/.ssh/known_hosts`). Після цього ви зможете встановити з'єднання з новим хостом.

Вхід на основі ключів

Ви можете налаштувати свій SSH-клієнт так, щоб він не надавав жодних паролів під час входу, а замість цього використовував відкриті ключі. Це найкращий метод підключення до віддаленого сервера через SSH, оскільки він набагато безпечніший. Перше, що вам потрібно зробити, це створити пару ключів на клієнтській машині. Для цього ви скористаєтеся `ssh-keygen` з опцією `-t`, яка вказує потрібний тип шифрування (*алгоритм цифрового підпису еліптичної кривої* у нашому випадку). Потім вас попросять вказати шлях для збереження пари ключів (`~/.ssh/` це зручно, а також є розташуванням за замовчуванням) і парольну фразу. Хоча парольна фраза необов'язкова, настійно рекомендується завжди її використовувати.

```
carol@debian:~/.ssh$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/carol/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/carol/.ssh/id_ecdsa.
Your public key has been saved in /home/carol/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:tlamD0SaTquPZYdNepwj8XN4xvqmHCbe8g5FKKUfMo8 carol@debian
The key's randomart image is:
+---[ECDSA 256]---+
|      .          |
|     o .         |
|    = o o        |
|     B *         |
|    E B S o      |
```

```
|     o & O       |
|      @ ^ =      |
|     *.@ @.      |
|    o.o+B+o      |
+----[SHA256]-----+
```

NOTE Під час створення пари ключів ви можете передати `ssh-keygen` параметр `-b`, щоб вказати розмір ключа в бітах (наприклад, `ssh-keygen -t ecdsa -b 521`).

Попередня команда створила ще два файли у вашому каталозі `~/.ssh`:

```
carol@debian:~/.ssh$ ls
id_ecdsa  id_ecdsa.pub  known_hosts
```

`id_ecdsa`

Це ваш закритий ключ.

`id_ecdsa.pub`

Це ваш відкритий ключ.

NOTE В асиметричній криптографії (так званій криптографії з відкритим ключем) відкритий і закритий ключі математично пов'язані один з одним таким чином, що все, що зашифровано одним, може бути розшифровано лише другим з цієї пари.

Наступне, що вам потрібно зробити, це додати свій відкритий ключ до файлу `~/.ssh/authorized_keys` користувача, під яким ви бажаєте увійти на віддаленому хості (якщо каталог `~/.ssh` ще не існує, вам доведеться спочатку його створити). Ви можете скопіювати свій відкритий ключ на віддалений сервер декількома способами: за допомогою флеш-пам'яті USB, за допомогою команди `scp`, яка передасть файл за допомогою SSH, або шляхом *вилучення* вмісту вашого відкритого ключа та передачі його в `ssh` таким чином:

```
carol@debian:~/.ssh$ cat id_ecdsa.pub |ssh ina@192.168.1.77 'cat >> .ssh/authorized_keys'
Password:
```

Коли ваш відкритий ключ буде додано до файлу `authorized_keys` на віддаленому хості, ви можете зіткнутися з двома сценаріями під час спроби встановити нове з'єднання:

- Якщо ви не вказали парольну фразу під час створення пари ключів, ви ввійдете в систему автоматично. Хоча цей метод зручний, він може бути небезпечним залежно від ситуації:

```
carol@debian:~$ ssh ina@192.168.1.77
Last login: Thu Jun 25 20:31:03 2020 from 192.168.1.55
Have a lot of fun...
ina@halof:~>
```

- Якщо ви вказали парольну фразу під час створення пари ключів, вам доведеться вводити її під час кожного з'єднання так само, якби це був пароль. Окрім відкритого ключа, цей метод додає додатковий рівень безпеки у вигляді парольної фрази, і тому його можна вважати більш безпечним. Що стосується зручності – однак – це точно так само, як необхідність вводити пароль кожного разу, коли ви встановлюєте з'єднання. Якщо ви не використовуєте парольну фразу і комусь вдасться отримати ваш приватний файл ключа SSH, вони матимуть доступ до кожного сервера, на якому встановлено ваш відкритий ключ.

```
carol@debian:~/.ssh$ ssh ina@192.168.1.77
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':
Last login: Thu Jun 25 20:39:30 2020 from 192.168.1.55
Have a lot of fun...
ina@halof:~>
```

Однак існує спосіб, який поєднує безпеку та зручність: використання *агента автентифікації SSH* (`ssh-agent`). Агент автентифікації має створити власну оболонку та зберігатиме ваші приватні ключі — для автентифікації відкритих ключів — у пам'яті до кінця сеансу. Давайте розглянемо, як це працює, трохи докладніше:

1. Використайте `ssh-agent`, щоб запустити нову оболонку Bash:

```
carol@debian:~/.ssh$ ssh-agent /bin/bash
carol@debian:~/.ssh$
```

2. Використайте команду `ssh-add`, щоб додати свій закритий ключ до безпечної області пам'яті. Якщо ви вказали парольну фразу під час генерації пари ключів (що рекомендовано для додаткової безпеки), ви отримаєте запит ввести парольну фразу:

```
carol@debian:~/.ssh$ ssh-add
```

```
Enter passphrase for /home/carol/.ssh/id_ecdsa:
Identity added: /home/carol/.ssh/id_ecdsa (carol@debian)
```

Після того, як вашу особу було додано, ви можете увійти на будь-який віддалений сервер, на якому присутній ваш відкритий ключ, без повторного введення парольної фрази. На сучасних комп'ютерах зазвичай виконується ця команда під час завантаження комп'ютера, оскільки вона залишатиметься в пам'яті, доки комп'ютер не вимкнеться (або ключ не буде вивантажено вручну).

Давайте завершимо цей розділ переліком чотирьох типів алгоритмів відкритого ключа, які можна вказати за допомогою `ssh-keygen`:

`RSA`

Названий на честь своїх творців Рона Ріввеста, Аді Шаміра та Леонарда Адлемана, він був опублікований у 1977 році. Він вважається безпечним і широко використовується досі. Його мінімальний розмір ключа становить 1024 біти (за замовчуванням 2048).

`DSA`

Алгоритм цифрового підпису виявився небезпечним, і вважається застарілим із OpenSSH 7.0. Ключі DSA повинні мати довжину точно 1024 біти.

`ecdsa`

Алгоритм цифрового підпису еліптичної кривої є вдосконаленням DSA і тому вважається більш безпечним. Він використовує еліптичну криптографію. Довжина ключа ECDSA визначається одним із трьох можливих розмірів еліптичної кривої в бітах: 256, 384 або 521.

`ed25519`

Це реалізація `EdDSA` (*алгоритму цифрового підпису кривої Едвардса*) який використовує сильнішу криву 25519. Він вважається найбезпечнішим з усіх. Усі ключі Ed25519 мають фіксовану довжину 256 біт.

NOTE Якщо викликати без специфікації `-t`, `ssh-keygen` за замовчуванням генеруватиме пару ключів `RSA`.

Роль ключів хоста сервера OpenSSH

Глобальний каталог конфігурації для OpenSSH знаходиться в каталозі `/etc`:

```
halof:~ # tree /etc/ssh
```

```
/etc/ssh
├── moduli
├── ssh_config
├── ssh_host_dsa_key
├── ssh_host_dsa_key.pub
├── ssh_host_ecdsa_key
├── ssh_host_ecdsa_key.pub
├── ssh_host_ed25519_key
├── ssh_host_ed25519_key.pub
├── ssh_host_rsa_key
├── ssh_host_rsa_key.pub
└── sshd_config

0 directories, 11 files
```

Окрім `moduli` та файлів конфігурації для клієнта (`ssh_config`) і сервера (`sshd_config`), ви знайдете чотири пари ключів — пару ключів для кожного підтримуваного алгоритму — які створюються під час встановлення сервера *OpenSSH*. Як уже зазначалося, сервер використовує ці *ключі хоста*, щоб ідентифікувати себе клієнтам за потреби. Схема їх назв така:

Закриті ключі

`ssh_host_` prefix + *algorithm* + `key` suffix (наприклад, `ssh_host_rsa_key`)

Відкриті ключі (або відбитки відкритих ключів)

`ssh_host_` prefix + *algorithm* + `key.pub` suffix (наприклад, `ssh_host_rsa_key.pub`)

NOTE

Відбиток (fingerprint) створюється шляхом застосування криптографічної хеш-функції до відкритого ключа. Оскільки відбитки коротші за ключі, до яких вони відносяться, вони стають в нагоді для спрощення певних завдань керування ключами.

Дозволи для файлів, що містять приватні ключі - це `0600` або `-rw-------`: тільки для читання та запису власником (`root`). З іншого боку, усі файли відкритих ключів також доступні для читання членам групи власників і всім іншим (`0644` або `-rw-r–r--`):

```
halof:~ # ls -l /etc/ssh/ssh_host_*
-rw------- 1 root root 1381 Dec 21 20:35 /etc/ssh/ssh_host_dsa_key
-rw-r--r-- 1 root root  605 Dec 21 20:35 /etc/ssh/ssh_host_dsa_key.pub
-rw------- 1 root root  505 Dec 21 20:35 /etc/ssh/ssh_host_ecdsa_key
-rw-r--r-- 1 root root  177 Dec 21 20:35 /etc/ssh/ssh_host_ecdsa_key.pub
-rw------- 1 root root  411 Dec 21 20:35 /etc/ssh/ssh_host_ed25519_key
```

```
-rw-r--r-- 1 root root   97 Dec 21 20:35 /etc/ssh/ssh_host_ed25519_key.pub
-rw------- 1 root root 1823 Dec 21 20:35 /etc/ssh/ssh_host_rsa_key
-rw-r--r-- 1 root root  397 Dec 21 20:35 /etc/ssh/ssh_host_rsa_key.pub
```

Ви можете переглянути відбитки ключів, передавши `ssh-keygen` перемикач `-l`. Ви також повинні використати `-f`, щоб вказати шлях до ключового файлу:

```
halof:~ # ssh-keygen -l -f /etc/ssh/ssh_host_ed25519_key
256 SHA256:8cnPrinC49ZHc+/9Ai5pV+1JfZ4WBRZhd3rDOsc2zlA root@halof (ED25519)
halof:~ # ssh-keygen -l -f /etc/ssh/ssh_host_ed25519_key.pub
256 SHA256:8cnPrinC49ZHc+/9Ai5pV+1JfZ4WBRZhd3rDOsc2zlA root@halof (ED25519)
```

Щоб переглянути відбиток ключа, а також його випадкове зображення, просто додайте перемикач `-v` наступним чином:

```
halof:~ # ssh-keygen -lv -f /etc/ssh/ssh_host_ed25519_key.pub
256 SHA256:8cnPrinC49ZHc+/9Ai5pV+1JfZ4WBRZhd3rDOsc2zlA root@halof (ED25519)
+--[ED25519 256]--+
|              +oo|
|             .+o.|
|        .    ..E.|
|         + .  +.o|
|        S +  + *o|
|          ooo Oo=|
|     . . . =o+.==|
|      = o =oo o=o|
|     o.o +o+..o.+|
+----[SHA256]-----+
```

Тунелі SSH-портів

OpenSSH має дуже потужну функцію пересилання, за допомогою якої трафік на вихідному порту тунелюється і шифрується через процес SSH, який потім перенаправляє його на порт на хості призначення. Цей механізм відомий як *тунелювання портів* або *переадресація портів* і має такі важливі переваги:

- Дозволяє обходити брандмауери для доступу до портів на віддалених хостах.
- Дозволяє отримати доступ ззовні до хоста у вашій приватній мережі.
- Він забезпечує шифрування для всього обміну даними.

Грубо кажучи, ми можемо розрізняти локальне та віддалене тунелювання портів.

Тунель локального порту

Ви визначаєте порт локально для перенаправлення трафіку на хост призначення через процес SSH, який знаходиться між ними. Процес SSH може виконуватися на локальному хості або на віддаленому сервері. Наприклад, якщо з якоїсь причини ви хочете тунелювати з'єднання з `www.gnu.org` через SSH за допомогою порту `8585` на вашій локальній машині, ви повинні зробити щось подібне до цього:

```
carol@debian:~$ ssh -L 8585:www.gnu.org:80 debian
carol@debian's password:
Linux debian 4.19.0-9-amd64 #1 SMP Debian 4.19.118-2 (2020-04-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
(...)
Last login: Sun Jun 28 13:47:27 2020 from 127.0.0.1
```

Пояснення таке: за допомогою перемикача `-L` ми вказуємо локальний порт `8585` для підключення до *http* порту `80` на `www.gnu.org` за допомогою процесу SSH, який працює на `debian` — нашому *localhost*. Ми могли б написати `ssh -L 8585:www.gnu.org:80 localhost` з таким же ефектом. Якщо ви зараз використовуєте веб-браузер для переходу на `http://localhost:8585`, вас буде перенаправлено на `www.gnu.org`. Для демонстрації ми будемо використовувати `lynx` (класичний текстовий веб-браузер):

```
carol@debian:~$ lynx http://localhost:8585
(...)
  * Back to Savannah Homepage
     * Not Logged in
     * Login
     * New User
     * This Page
     * Language
     * Clean Reload
     * Printer Version
     * Search
     * _
(...)
```

Якби ви хотіли зробити те саме, але підключилися через процес SSH, що працює на `halof`, ви б діяли так:

```
carol@debian:~$ ssh -L 8585:www.gnu.org:80 -Nf ina@192.168.1.77
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':
carol@debian:~$
carol@debian:~$ lynx http://localhost:8585
(...)
  * Back to Savannah Homepage
     * Not Logged in
     * Login
     * New User
     * This Page
     * Language
     * Clean Reload
     * Printer Version
     * Search
     * _
(...)
```

Важливо звернути увагу на три деталі команди:

- Завдяки опції `-N` ми не входили в `halof`, а замість цього виконали переадресацію портів.
- Опція `-f` вказала SSH працювати у фоновому режимі.
- Ми вказали користувача `ina` для переадресації: `ina@192.168.1.77`

Тунель віддаленого порту

Під час тунелювання віддаленого порту (або зворотного перенаправлення портів) трафік, що надходить на порт на віддаленому сервері, перенаправляється до процесу SSH, який працює на вашому локальному хості, а звідти на вказаний порт на сервері призначення (який також може бути вашою локальною машиною). Наприклад, скажімо, ви хочете дозволити комусь із-за меж вашої мережі отримати доступ до веб-сервера Apache, який працює на вашому локальному хості, через порт `8585` SSH-сервера, який працює на `halof` (`192.168.1.77`). Ви б застосували таку команду:

```
carol@debian:~$ ssh -R 8585:localhost:80 -Nf ina@192.168.1.77
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':
carol@debian:~$
```

Тепер кожен, хто встановить з'єднання з `halof` на порту `8585`, побачить домашню сторінку `Debian`'s Apache2 за замовчуванням:

```
carol@debian:~$ lynx 192.168.1.77:8585
(...)
                                                                Apache2 Debian Default
Page: It works (p1 of 3)
   Debian Logo Apache2 Debian Default Page
   It works!

   This is the default welcome page used to test the correct operation of the Apache2 server
after
   installation on Debian systems. If you can read this page, it means that the Apache HTTP
server
   installed at this site is working properly. You should replace this file (located at
   /var/www/html/index.html) before continuing to operate your HTTP server.
(...)
```

NOTE Існує третій, більш складний тип переадресації портів, який виходить за рамки цього уроку: *динамічна переадресація портів*. Замість взаємодії з одним портом цей тип переадресації використовує різні зв'язки TCP через діапазон портів.

Тунелі X11

Тепер, коли ви розумієте тунелі портів, давайте завершимо цей урок обговоренням тунелювання X11 (також відомого як *X11forwarding*). Через тунель X11 *X Window System* на віддаленому хості пересилається на вашу локальну машину. Для цього вам просто потрібно передати `ssh` параметр `-X`:

```
carol@debian:~$ ssh -X ina@halof
...
```

Тепер ви можете запустити графічну програму, таку як веб-браузер `firefox`, з таким результатом: програма буде запущена на віддаленому сервері, але її відображення буде переадресовано на ваш локальний хост.

Якщо замість цього розпочати новий сеанс SSH із опцією `-x`, *X11forwarding* буде вимкнено. Спробуйте запустити `firefox` зараз, і ви отримаєте помилку на зразок такої:

```
carol@debian:~$ ssh -x ina@halof
carol@192.168.0.106's password:
(...)
```

```
ina@halof:~$ firefox

(firefox-esr:1779): Gtk-WARNING **: 18:45:45.603: Locale not supported by C library.
    Using the fallback 'C' locale.
Error: no DISPLAY environment variable specified
```

NOTE

Три директиви конфігурації, пов'язані з переадресацією локального порту, переадресацією віддаленого порту та переадресацією X11, це `AllowTcpForwarding`, `GatewayPorts` і `X11Forwarding` відповідно. Для отримання додаткової інформації введіть `man ssh_config` та/або `man sshd_config`.

Вправи до посібника

1. Увійшовши як користувач `sonya` на вашій клієнтській машині, виконайте такі завдання SSH на віддаленому сервері `halof`:
   - Виконайте команду для отримання списку вмісту `~/.ssh` від імені користувача `serena` на віддаленому хості; потім поверніться до свого локального терміналу.
   - Увійдіть як користувач `serena` на віддаленому хості.
   - Увійдіть як користувач `sonya` на віддаленому хості.
   - Видаліть усі ключі, що належать `halof`, із локального файлу `~/.ssh/known_hosts`.
   - На клієнтській машині створіть пару ключів `ecdsa` з 256 біт.
   - На вашому клієнтському комп'ютері створіть пару ключів `ed25519` з 256 біт.
2. Виконайте наступні кроки в правильному порядку, щоб встановити SSH-з'єднання за допомогою *агента автентифікації SSH*:
   - На клієнті запустіть нову оболонку Bash для *агента автентифікації* за допомогою `ssh-agent /bin/bash`.
   - На клієнті створіть пару ключів за допомогою `ssh-keygen`.
   - На клієнті додайте свій закритий ключ до безпечної області пам'яті за допомогою `ssh-add`.
   - Додайте відкритий ключ вашого клієнта до файлу `~/.ssh/authorized_keys` користувача, як якого ви хочете увійти на віддаленому хості.
   - Якщо він ще не існує, створіть `~/.ssh` для користувача, під яким ви бажаєте увійти на сервер.
   - Підключіться до віддаленого сервера.

   Правильний порядок:

Крок 1:	
Крок 2:	
Крок 3:	
Крок 4:	
Крок 5:	
Крок 6:	

3. Що стосується *переадресації портів*, які параметри та директиви використовуються для таких типів тунелів:

Тип тунелю	**Варіант**	**Директива**
Локальний		
Віддалений або зворотний		
X		

4. Припустимо, ви вводите команду `ssh -L 8888:localhost:80 -Nf ina@halof` у терміналі вашого клієнтського комп'ютера. Перебуваючи на клієнтській машині, ви вказуєте браузер на `http://localhost:8888`. Що ви отримаєте?

Дослідницькі вправи

1. Щодо директив безпеки SSH:
   - Яка директива використовується в `/etc/ssh/sshd_config` для ввімкнення `root` входу:
   - Яку директиву ви б використали в `/etc/ssh/sshd_config`, щоб вказати лише локальний обліковий запис для SSH-підключення:
2. Якщо ви використовуєте одного користувача як на клієнті, так і на сервері, яку команду `ssh` ви можете використати, щоб передати відкритий ключ клієнта на сервер, щоб ви могли ввійти через автентифікацію відкритого ключа?
3. Створіть два тунелі локальних портів однією командою, перенаправляючи локальні непривілейовані порти 8080 і 8585 через віддалений сервер `halof` на веб-сайти `www.gnu.org` і `www.melpa.org` відповідно. Використовуйте користувача `ina` на віддаленому сервері та не забудьте використовувати перемикачі `-Nf`:

Підсумки

У цьому уроці ми обговорили *OpenSSH 2*, який використовує протокол *Secure Shell* для шифрування зв'язку між сервером і клієнтом. Ви дізналися:

- Як увійти на віддалений сервер.
- Як виконувати команди віддалено.
- Як створювати пари ключів.
- Як встановити вхід на основі ключів.
- Як використовувати *агент автентифікації* для додаткової безпеки та зручності.
- Алгоритми відкритого ключа, які підтримуються *OpenSSH*: `RSA`, `DSA`, `ecdsa`, `ed25519`.
- Роль ключів хоста *OpenSSH*.
- Як створити тунелі портів: локальні, віддалені та X.

У цьому уроці обговорювалися такі команди:

`ssh`
: Вхід або виконання команд на віддаленій машині.

`ssh-keygen`
: Створення, керування та перетворення ключів автентифікації.

`ssh-agent`
: Агент автентифікації OpenSSH.

`ssh-add`
: Додавання ідентифікаторів приватного ключа до агента автентифікації.

Відповіді до вправ посібника

1. Увійшовши як користувач `sonya` на вашій клієнтській машині, виконайте такі завдання SSH на віддаленому сервері `halof`:
   - Виконайте команду для отримання списку вмісту `~/.ssh` від імені користувача `serena` на віддаленому хості; потім поверніться до свого локального терміналу.

     ```
     ssh serena@halof ls .ssh
     ```

   - Увійдіть як користувач `serena` на віддаленому хості.

     ```
     ssh serena@halof
     ```

   - Увійдіть як користувач `sonya` на віддаленому хості.

     ```
     ssh halof
     ```

   - Видаліть усі ключі, що належать `halof`, із локального файлу `~/.ssh/known_hosts`.

     ```
     ssh-keygen -R halof
     ```

   - На клієнтській машині створіть пару ключів `ecdsa` з 256 біт.

     ```
     ssh-keygen -t ecdsa -b 256
     ```

   - На вашому клієнтському комп'ютері створіть пару ключів `ed25519` з 256 біт.

     ```
     ssh-keygen -t ed25519
     ```

2. Виконайте наступні кроки в правильному порядку, щоб встановити SSH-з'єднання за допомогою *агента автентифікації SSH*:
   - На клієнті запустіть нову оболонку Bash для *агента автентифікації* за допомогою `ssh-agent /bin/bash`.
   - На клієнті створіть пару ключів за допомогою `ssh-keygen`.
   - На клієнті додайте свій закритий ключ до безпечної області пам'яті за допомогою

`ssh-add`.

- Додайте відкритий ключ вашого клієнта до файлу `~/.ssh/authorized_keys` користувача, як якого ви хочете увійти на віддаленому хості.
- Якщо він ще не існує, створіть `~/.ssh` для користувача, під яким ви бажаєте увійти на сервер.
- Підключіться до віддаленого сервера.

Правильний порядок:

Крок 1:	На клієнті створіть пару ключів за допомогою `ssh-keygen`.
Крок 2:	Якщо він ще не існує, створіть `~/.ssh` для користувача, від імені якого ви бажаєте увійти на сервер.
Крок 3:	Додайте відкритий ключ вашого клієнта до файлу `~/.ssh/authorized_keys` користувача, під яким ви хочете увійти на віддаленому хості.
Крок 4:	На клієнті запустіть нову оболонку Bash для *агента автентифікації* за допомогою `ssh-agent /bin/bash`.
Крок 5:	На клієнті додайте свій закритий ключ до безпечної області пам'яті за допомогою `ssh-add`.
Крок 6:	Підключіться до віддаленого сервера.

3. Що стосується *переадресації портів*, які параметри та директиви використовуються для таких типів тунелів:

Тип тунелю	**Варіант**	**Директива**
Локальний	`-L`	`AllowTcpForwarding`
Віддалений або зворотний	`-R`	`GatewayPorts`
X	`-X`	`X11Forwarding`

4. Припустимо, ви вводите команду `ssh -L 8888:localhost:80 -Nf ina@halof` у терміналі вашого клієнтського комп'ютера. Перебуваючи на клієнтській машині, ви

вказуєте браузер на `http://localhost:8888`. Що ви отримаєте?

Домашню сторінка веб-сервера `halof`, надану сервером як ресурс `localhost`.

Відповіді до дослідницьких вправ

1. Щодо директив безпеки SSH:
   - Яка директива використовується в `/etc/ssh/sshd_config` для ввімкнення `root` входу:

     `PermitRootLogin`

   - Яку директиву ви б використали в `/etc/ssh/sshd_config`, щоб вказати лише локальний обліковий запис для SSH-підключення:

     `AllowUsers`

2. Якщо ви використовуєте одного користувача як на клієнті, так і на сервері, яку команду `ssh` ви можете використати, щоб передати відкритий ключ клієнта на сервер, щоб ви могли ввійти через автентифікацію відкритого ключа?

   `ssh-copy-id`

3. Створіть два тунелі локальних портів однією командою, перенаправляючи локальні непривілейовані порти 8080 і 8585 через віддалений сервер `halof` на веб-сайти `www.gnu.org` і `www.melpa.org` відповідно. Використовуйте користувача `ina` на віддаленому сервері та не забудьте використовувати перемикачі `-Nf`:

   ```
   ssh -L 8080:www.gnu.org:80 -L 8585:www.melpa.org:80 -Nf ina@halof
   ```

110.3 Урок 2

Сертифікат:	LPIC-1
Версія:	5.0
Розділ:	110 Безпека
Тема:	110.3 Захист даних за допомогою шифрування
Урок:	2 з 2

Вступ

У попередньому уроці ми дізналися, як використовувати *OpenSSH* для шифрування сеансів віддаленого входу, а також будь-якого іншого подальшого обміну інформацією. Можуть бути й інші сценарії, коли вам може знадобитися зашифрувати файли чи електронну пошту, щоб вони безпечно були доставлені одержувачам і не потрапили на сторонні очі. Вам також може знадобитися цифровий підпис цих файлів або повідомлень, щоб запобігти їх фальсифікації.

Чудовим інструментом для таких видів використання є *GNU Privacy Guard* (він же *GnuPG* або просто *GPG*), який є безкоштовною реалізацією з відкритим вихідним кодом пропрієтарної *Pretty Good Privacy* (*PGP*). *GPG* використовує стандарт *OpenPGP*, як визначено *Робочою групою _OpenPGP Internet Engineering Task Force* (*IETF*) у RFC 4880. У цьому уроці ми розглянемо основи *GNU Privacy Guard.*

Виконання базової конфігурації, використання та відкликання GnuPG

Як і у випадку з SSH, основним механізмом GPG є *асиметрична криптографія* або *криптографія з відкритим ключем*. Користувач генерує пару ключів, яка складається з *закритого ключа* та *відкритого ключа*. Ключі математично пов'язані таким чином, що те, що зашифровано одним, може бути розшифровано лише іншим з цієї пари. Для успішного спілкування користувач повинен надіслати свій відкритий ключ одержувачу.

Конфігурація та використання GnuPG

Команда для роботи з GPG – `gpg`. Ви можете передати їй ряд варіантів для виконання різних завдань. Давайте почнемо з генерації пари ключів для користувача `carol`. Для цього ви скористаєтеся командою `gpg --gen-key`:

```
carol@debian:~$ gpg --gen-key
gpg (GnuPG) 2.2.12; Copyright (C) 2018 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory '/home/carol/.gnupg' created
gpg: keybox '/home/carol/.gnupg/pubring.kbx' created
Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name:

(...)
```

Після того, як ви повідомите, серед іншого, що каталог конфігурації `~/.gnupg` і ваш публічний ключ `~/.gnugpg/pubring.kbx` створено, `gpg` просить вас надати ваше справжнє ім'я та електронну адресу:

```
(...)
Real name: carol
Email address: carol@debian
You selected this USER-ID:
    "carol <carol@debian>"
```

```
Change (N)ame, (E)mail, or (O)kay/(Q)uit?
```

Якщо ви погоджуєтеся з `USER-ID` і натиснете `O`, вас запитають про пароль (бажано, щоб він був достатньо складним):

```
┌──────────────────────────────────────────────────────┐
│ Please enter the passphrase to                       │
│ protect your new key                                 │
│                                                      │
│ Passphrase:  │

(...)
```

Буде відображено кілька останніх повідомлень про створення інших файлів, а також самих ключів, після чого ви завершите процес генерації ключів:

```
(...)
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /home/carol/.gnupg/trustdb.gpg: trustdb created
gpg: key 19BBEFD16813034E marked as ultimately trusted
gpg: directory '/home/carol/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/carol/.gnupg/openpgp-
revocs.d/D18FA0021F644CDAF57FD0F919BBEFD16813034E.rev'
public and secret key created and signed.

pub   rsa3072 2020-07-03 [SC] [expires: 2022-07-03]
      D18FA0021F644CDAF57FD0F919BBEFD16813034E
uid                      carol <carol@debian>
sub   rsa3072 2020-07-03 [E] [expires: 2022-07-03]
```

Тепер ви можете побачити, що знаходиться всередині каталогу `~/.gnupg` (каталог конфігурації GPG):

```
carol@debian:~/.gnupg$ ls -l
total 16
drwx------ 2 carol carol 4096 Jul  3 23:34 openpgp-revocs.d
drwx------ 2 carol carol 4096 Jul  3 23:34 private-keys-v1.d
-rw-r--r-- 1 carol carol 1962 Jul  3 23:34 pubring.kbx
```

```
-rw------- 1 carol carol 1240 Jul  3 23:34 trustdb.gpg
```

Пояснимо призначення кожного файлу:

`opengp-revocs.d`

Тут зберігається сертифікат відкликання, створений разом із парою ключів. Дозволи для цього каталогу досить обмежені, оскільки кожен, хто має доступ до сертифіката, може відкликати ключ (докладніше про відкликання ключа в наступному підрозділі).

`private-keys-v1.d`

Це каталог, у якому зберігаються ваші приватні ключі, тому дозволи є обмежувальними.

`pubring.kbx`

Це ваше сховище публічний ключів. Воно зберігає ваші власні та будь-які інші імпортовані відкриті ключі.

`trustdb.gpg`

База даних довіри. Це пов'язано з концепцією *Web of Trust* (яка виходить за рамки цього уроку).

NOTE Поява *GnuPG 2.1* принесла деякі значні зміни, наприклад, зникнення файлів `secring.gpg` і `pubring.gpg` на користь `private-keys-v1.d` і `pubring.kbx`, відповідно.

Після створення вашої пари ключів ви можете переглянути свої відкриті ключі за допомогою команди `gpg --list-keys` — яка відобразить вміст вашого сховища публічних ключів:

```
carol@debian:~/.gnupg$ gpg --list-keys
/home/carol/.gnupg/pubring.kbx
------------------------------
pub   rsa3072 2020-07-03 [SC] [expires: 2022-07-03]
      D18FA0021F644CDAF57FD0F919BBEFD16813034E
uid           [ultimate] carol <carol@debian>
sub   rsa3072 2020-07-03 [E] [expires: 2022-07-03]
```

Шістнадцятковий рядок `D18FA0021F644CDAF57FD0F919BBEFD16813034E` — це ваш *відбиток відкритого ключа.*

NOTE Окрім `USER-ID` (`carol` у прикладі), є також `KEY-ID`. `KEY-ID` складається з

останніх 8 шістнадцяткових цифр вашого відбитка відкритого ключа (`6813 034E`). Ви можете перевірити відбиток ключа за допомогою команди `gpg --fingerprint USER-ID`.

Розповсюдження та відкликання ключів

Тепер, коли у вас є відкритий ключ, вам слід зберегти його (тобто *експортувати*) у файл, щоб ви могли зробити його доступним для своїх майбутніх одержувачів. Потім вони зможуть використовувати його для шифрування файлів або повідомлень, призначених для вас (оскільки ви єдиний, хто володіє закритим ключем, також тільки ви зможете розшифрувати та прочитати їх). Подібним чином, ваші одержувачі також використовуватимуть його для розшифровки та перевірки ваших зашифрованих або підписаних повідомлень/файлів. Використовуйте команду `gpg --export`, за якою слідує `USER-ID` і перенаправлення до імені вихідного файлу за вашим вибором:

```
carol@debian:~/.gnupg$ gpg --export carol > carol.pub.key
carol@debian:~/.gnupg$ ls
carol.pub.key  openpgp-revocs.d  private-keys-v1.d  pubring.kbx  trustdb.gpg
```

NOTE

Передача опції `-a` або `--armor` до `gpg --export`(наприклад, `gpg --export --armor carol > carol.pub.key`) створить захищені вихідні дані формату ASCII (замість типового двійкового формату OpenPGP), які можна безпечно надсилати електронною поштою.

Як уже зазначалося, тепер ви повинні надіслати свій файл відкритого ключа (`carol.pub.key`) одержувачу, з яким ви хочете обмінятися інформацією. Наприклад, давайте надішлемо файл відкритого ключа до `ina` на віддаленому сервері `halof` за допомогою `scp`(*secure copy*):

```
carol@debian:~/.gnupg$ scp carol.pub.key ina@halof:/home/ina/
Enter passphrase for key '/home/carol/.ssh/id_ecdsa':
carol.pub.key
100% 1740   775.8KB/s   00:00
carol@debian:~/.gnupg$
```

`ina` тепер належить `carol.pub.key`. Вона використає його для шифрування файлу та надішле його до `carol` у наступній секції.

NOTE

Іншим способом розповсюдження відкритих ключів є використання *серверів ключів*: ви завантажуєте свій відкритий ключ на сервер за допомогою

> команди `gpg --keyserver keyserver-name --send-keys KEY-ID`, а інші користувачі отримають (тобто *import*) їх за допомогою `gpg --keyserver keyserver-name --recv-keys KEY-ID`.

Давайте завершимо цей розділ обговоренням відкликання ключа. Відкликання ключа слід використовувати, якщо ваші приватні ключі скомпрометовано або вилучено з експлуатації. Першим кроком є створення сертифіката відкликання, передавши `gpg` параметр `--gen-revoke`, а потім `USER-ID`. Ви можете перед `--gen-revoke` вказати параметр `--output`, а потім вказати ім'я цільового файлу, щоб зберегти отриманий сертифікат у файлі (замість того, щоб він виводився на екран терміналу). Вихідні повідомлення під час процесу відкликання досить зрозумілі:

```
sonya@debian:~/.gnupg$ gpg --output revocation_file.asc --gen-revoke sonya

sec  rsa3072/0989EB7E7F9F2066 2020-07-03 sonya <sonya@debian>

Create a revocation certificate for this key? (y/N) y
Please select the reason for the revocation:
  0 = No reason specified
  1 = Key has been compromised
  2 = Key is superseded
  3 = Key is no longer used
  Q = Cancel
(Probably you want to select 1 here)
Your decision? 1
Enter an optional description; end it with an empty line:
> My laptop was stolen.
>
Reason for revocation: Key has been compromised
My laptop was stolen.
Is this okay? (y/N) y
ASCII armored output forced.
Revocation certificate created.

Please move it to a medium which you can hide away; if Mallory gets
access to this certificate he can use it to make your key unusable.
It is smart to print this certificate and store it away, just in case
your media become unreadable.  But have some caution:  The print system of
your machine might store the data and make it available to others!
```

Сертифікат відкликання збережено у файлі `revocation_file.asc` (`asc` для формату ASCII):

```
sonya@debian:~/.gnupg$ ls
openpgp-revocs.d  private-keys-v1.d  pubring.kbx  revocation_file.asc  trustdb.gpg
sonya@debian:~/.gnupg$ cat revocation_file.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: This is a revocation certificate

iQHDBCABCgAtFiEEiIVjfDnnpieFi0wvnlcN6yLCeHEFAl8ASx4PHQJzdG9sZW4g
bGFwdG9wAAoJEJ5XDesiwnhxT9YMAKkjQiMpo9Uyiy9hyvukPPSrLcmtAGLk4pKS
pLZfzA5kxa+HPQwBglAEvfNRR6VMxqXUgUGYC/IAyQQM62oNAcY2PCPrxyJNgVF7
8l4mMZKvW++5ikjZwyg6WWV0+w6oroeo9qruJFjcu752p4T+9gsHVa2r+KRqcPQe
aZ65sAvsBJlcsUDZqfWUXg2kQp9mNPCdQuqvDaKRgNCHA1zbzNFzXWVd2X5RgFo5
nY+tUP8ZQA9DTQPBLPcggICmfLopMPZYB2bft5geb2mMi2oNpf9CNPdQkdccimNV
aRjqdUP9C89PwTafBQkQiONlsR/dWTFcqprG5KOWQPA7xjeMV8wretdEgsyTxqHp
v1iRzwjshiJCKBXXvz7wSmQrJ4OfiMDHeS4ipR0AYdO8QCzmOzmcFQKikGSHGMy1
z/YRlttd6NZIKjf1TD0nTrFnRvPdsZOlKYSArbfqNrHRBQkgirOD4JPI1tYKTffq
iOeZFx25K+fj2+0AJjvrbe4HDo5m+Q==
=umI8
-----END PGP PUBLIC KEY BLOCK-----
```

Щоб ефективно відкликати свій приватний ключ, тепер вам потрібно об'єднати сертифікат із ключем, що робиться шляхом імпорту файлу сертифіката відкликання до зв'язки ключів:

```
sonya@debian:~/.gnupg$ gpg --import revocation_file.asc
gpg: key 9E570DEB22C27871: "sonya <sonya@debian>" revocation certificate imported
gpg: Total number processed: 1
gpg:    new key revocations: 1
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2022-07-04
```

Виведіть список своїх ключів зараз, і ви отримаєте інформацію про ваш відкликаний ключ:

```
sonya@debian:~/.gnupg$ gpg --list-keys
/home/sonya/.gnupg/pubring.kbx
pub   rsa3072 2020-07-04 [SC] [revoked: 2020-07-04]
      8885637C39E7A627858B4C2F9E570DEB22C27871
uid           [ revoked] sonya <sonya@debian>
```

Нарешті, що не менш важливо, переконайтеся, що ви зробили відкликаний ключ

доступним для будь-якої сторони, яка має пов'язані з ним публічні ключі (включно з серверами ключів).

Використовуйте GPG для шифрування, дешифрування, підпису та перевірки файлів

У попередньому розділі `carol` надіслала свій відкритий ключ `ina`. Зараз ми використаємо його, щоб обговорити, як GPG може шифрувати, розшифровувати, підписувати та перевіряти файли.

Шифрування та дешифрування файлів

По-перше, `ina` має імпортувати відкритий ключ `carol` (`carol.pub.key`) для свого сховища ключів, щоб вона могла почати з ним працювати:

```
ina@halof:~> gpg --import carol.pub.key
gpg: /home/ina/.gnupg/trustdb.gpg: trustdb created
gpg: key 19BBEFD16813034E: public key "carol <carol@debian>" imported
gpg: Total number processed: 1
gpg:               imported: 1
ina@halof:~> gpg --list-keys
/home/ina/.gnupg/pubring.kbx
----------------------------
pub   rsa3072 2020-07-03 [SC] [expires: 2022-07-03]
      D18FA0021F644CDAF57FD0F919BBEFD16813034E
uid           [ unknown] carol <carol@debian>
sub   rsa3072 2020-07-03 [E] [expires: 2022-07-03]
```

Далі ви створите файл, вписавши в нього певний текст, а потім зашифруєте його за допомогою `gpg` (оскільки ви не підписали ключ `carol`), вас явно запитають, чи хочете ви використовувати цей ключ):

```
ina@halof:~> echo "This is the message ..." > unencrypted-message
ina@halof:~> gpg --output encrypted-message --recipient carol --armor --encrypt unencrypted-
message
gpg: 0227347CC92A5CB1: There is no assurance this key belongs to the named user
sub  rsa3072/0227347CC92A5CB1 2020-07-03 carol <carol@debian>
 Primary key fingerprint: D18F A002 1F64 4CDA F57F  D0F9 19BB EFD1 6813 034E
      Subkey fingerprint: 9D89 1BF9 39A4 C130 E44B  1135 0227 347C C92A 5CB1

It is NOT certain that the key belongs to the person named
```

```
in the user ID.  If you really know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
```

Розберемо команду `gpg`:

`--output encrypted-message`

Специфікація назви файлу для зашифрованої версії вихідного файлу (`encrypted-message` у прикладі).

`--recipient carol`

Специфікація `USER-ID` одержувача (`carol` у нашому прикладі). Якщо не вказано, GnuPG запитає його (якщо не вказано `--default-recipient`).

`--armor`

Цей параметр створює захищені вихідні дані ASCII, які можна скопіювати в електронний лист.

`--encrypt unencrypted-message`

Специфікація назви оригінального файлу для шифрування.

Тепер ви можете надіслати `encrypted-message` до `carol` на `debian` за допомогою `scp`:

```
ina@halof:~> scp encrypted-message carol@debian:/home/carol/
carol@debian's password:
encrypted-message                                                    100%  736
1.8MB/s   00:00
```

Якщо ви зараз увійдете як `carol` і спробуєте прочитати `encrypted-message`, ви переконаєтеся, що воно насправді зашифроване і, отже, прочитати його неможливо:

```
carol@debian:~$ cat encrypted-message
-----BEGIN PGP MESSAGE-----

hQGMAwInNHzJKlyxAQv/brJ8Ubs/xya35sbv6kdRKm1C7ONLxL3OueWA4mCs0Y/P
GBna6ZEUCrMEgl/rCyByj3Yq74kuiTmzxAIRUDdvHfj0TtrOWjVAqIn/fPSfMkjk
dTxKo1i55tLJ+sj17dGMZDcNBinBTP4U1atuN71A5w7vH+XpcesRcFQLKiSOmYTt
F7SN3/5x5J6io4ISn+b0KbJgiJNNx+Ne/ub4Uzk4NlK7tmBklyC1VRualtxcG7R9
1klBPYSld6fTdDwT1Y4MofpyILAiGMZvUR1RXauEKf7OIzwC5gWU+UQPSgeCdKQu
X7QL0ZIBS0Ug2XKrO1k93lmDjf8PWsRIml6n/hNelaOBA3HMP0b6Ozv1gFeEsFvC
```

```
IxhUYPb+rfuNFTMEB7xIO94AAmWB9N4qknMxdDqNE8WhA728Plw6y8L2ngsplY15
MR4lIFDpljA/CcVh4BXVe9j0TdFWDUkrFMfaIfcPQwKLXEYJp19XYIaaEazkOs5D
W4pENN0YOcX0KWyAYX6r0l8BF0rq/HMenQwqAVXMG3s8ATuUOeqjBbR1x1qCvRQP
CR/3V73aQwc2j5ioQmhWYpqxiro0yKX2Ar/E6rZyJtJYrq+CUk8O3JoBaudknNFj
pwuRwF1amwnSZ/MZ/9kMKQ==
=g1jw
-----END PGP MESSAGE-----
```

Однак, оскільки у вас є приватний ключ, ви можете легко розшифрувати повідомлення, передавши `gpg` опцію `--decrypt`, а потім шлях до зашифрованого файлу (потрібна буде парольна фраза приватного ключа):

```
carol@debian:~$ gpg --decrypt encrypted-message
gpg: encrypted with 3072-bit RSA key, ID 0227347CC92A5CB1, created 2020-07-03
      "carol <carol@debian>"
This is the message ...
```

Ви також можете вказати опцію `--output`, щоб зберегти повідомлення в новому незашифрованому файлі:

```
carol@debian:~$ gpg --output unencrypted-message --decrypt encrypted-message
gpg: encrypted with 3072-bit RSA key, ID 0227347CC92A5CB1, created 2020-07-03
      "carol <carol@debian>"
carol@debian:~$ cat unencrypted-message
This is the message ...
```

Підписання та перевірка файлів

Окрім шифрування, GPG також можна використовувати для підпису файлів. Тут актуальна опція `--sign`. Давайте почнемо зі створення нового повідомлення (`message`) і підпишемо його опцією `--sign` (потрібна буде парольна фраза вашого закритого ключа):

```
carol@debian:~$ echo "This is the message to sign ..." > message
carol@debian:~$ gpg --output message.sig --sign message
(...)
```

Складові команди `gpg`:

`--output message`

Специфікація імені файлу підписаної версії вихідного файлу (`message.sig` у нашому

прикладі).

`--sign message`

Шлях до оригінального файлу.

NOTE За допомогою `--sign` документ стискається, а потім підписується. Вихідні дані у двійковому форматі.

Далі ми передамо файл до `ina` на `halof` за допомогою `scp message.sig ina@halof:/home/ina`. Повернувшись до `ina` на `halof`, ви можете перевірити це за допомогою параметра `--verify`:

```
ina@halof:~> gpg --verify message.sig
gpg: Signature made Sat 04 jul 2020 14:34:41 CEST
gpg:                using RSA key D18FA0021F644CDAF57FD0F919BBEFD16813034E
gpg: Good signature from "carol <carol@debian>" [unknown]
(...)
```

Якщо ви також хочете прочитати файл, вам потрібно розшифрувати його в новий файл (у нашому випадку `message`) за допомогою параметра `--output`:

```
ina@halof:~> gpg --output message --decrypt message.sig
gpg: Signature made Sat 04 jul 2020 14:34:41 CEST
gpg:                using RSA key D18FA0021F644CDAF57FD0F919BBEFD16813034E
gpg: Good signature from "carol <carol@debian>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: D18F A002 1F64 4CDA F57F  D0F9 19BB EFD1 6813 034E
ina@halof:~> cat message
This is the message to sign ...
```

GPG-агент

Ми завершимо цей урок знайомством з `gpg-agent`. `gpg-agent` — це демон, який керує приватними ключами для GPG (він запускається на вимогу `gpg`). Щоб переглянути підсумок найбільш корисних параметрів, запустіть `gpg-agent --help` або `gpg-agent -h`:

```
carol@debian:~$ gpg-agent --help
gpg-agent (GnuPG) 2.2.4
libgcrypt 1.8.1
Copyright (C) 2017 Free Software Foundation, Inc.
```

```
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Syntax: gpg-agent [options] [command [args]]
Secret key management for GnuPG

Options:

     --daemon                      run in daemon mode (background)
     --server                      run in server mode (foreground)
     --supervised                  run in supervised mode
 -v, --verbose                     verbose
 -q, --quiet                       be somewhat more quiet
 -s, --sh                          sh-style command output
 -c, --csh                         csh-style command output
(...)
```

NOTE Для отримання додаткової інформації зверніться до man-сторінки `gpg-agent`.

Вправи до посібника

1. Заповніть таблицю, вказавши правильну назву файлу:

Опис	**Ім’я файлу**
Довірча база даних	
Довідник свідоцтв про відкликання	
Каталог закритих ключів	
Відкритий ключ	

2. Дай відповідь на наступні запитання:
   - Який тип криптографії використовується *GnuPG*?
   - Які два основні компоненти криптографії з відкритим ключем?
   - Що таке `KEY-ID` відбитка відкритого ключа `07A6 5898 2D3A F3DD 43E3 DA95 1F3F 3147 FA7F 54C7`?
   - Який метод використовується для розподілу відкритих ключів на глобальному рівні?

3. Розмістіть наступні кроки в правильному порядку щодо відкликання закритого ключа:
   - Надайте відкликаний ключ своїм кореспондентам.
   - Створіть сертифікат відкликання.
   - Імпортуйте сертифікат відкликання до своєї зв’язки ключів.

   Правильний порядок:

Крок 1:	
Крок 2:	
Крок 3:	

4. Стосовно шифрування файлів, що означає параметр `--armor` у команді `gpg --output encrypted-message --recipient carol --armor --encrypt unencrypted-message`?

Дослідницькі вправи

1. Більшість параметрів `gpg` мають як довгу, так і коротку версії. Доповніть таблицю відповідним коротким варіантом:

Довга версія	**Коротка версія**
`--armor`	
`--output`	
`--recipient`	
`--decrypt`	
`--encrypt`	
`--sign`	

2. Дайте відповідь на такі запитання щодо експорту ключів:
   - Яку команду ви використаєте, щоб експортувати всі свої відкриті ключі у файл під назвою `all.key`?
   - Яку команду ви використали б, щоб експортувати всі свої приватні ключі у файл під назвою `all_private.key`?

3. Який параметр `gpg` дозволяє виконувати більшість завдань, пов'язаних із керуванням ключами, показуючи вам меню?

4. Який параметр `gpg` дозволяє зробити цифровий підпис відкритого тексту?

Підсумки

Цей урок охопив *GNU Privacy Guard*, чудовий вибір для шифрування/дешифрування та цифрового підпису/перевірки файлів. Ви дізналися:

- Як згенерувати пару ключів.
- Як вивести список ключів з вашого сховищі.
- Вміст каталогу `~/.gnupg`.
- Що таке `USER-ID` і `KEY-ID`.
- Як роздавати відкриті ключі своїм кореспондентам.
- Як глобально поширювати відкриті ключі через сервери ключів.
- Як відкликати закриті ключі.
- Як шифрувати та розшифровувати файли.
- Як підписувати та перевіряти файли.
- Основне поняття *GPG-агенту*.

У цьому уроці обговорювалися такі команди:

`gpg`
: *OpenPGP* інструмент шифрування та підпису.

Відповіді до вправ посібника

1. Заповніть таблицю, вказавши правильну назву файлу:

Опис	**Ім’я файлу**
Довірча база даних	`trustdb.gpg`
Довідник свідоцтв про відкликання	`opengp-revocs.d`
Каталог закритих ключів	`private-keys-v1.d`
Відкритий ключ	`pubring.kbx`

2. Дай відповідь на наступні запитання:
   - Який тип криптографії використовується *GnuPG*?

     Криптографія з відкритим ключем або асиметрична криптографія.
   - Які два основні компоненти криптографії з відкритим ключем?

     Відкритий і закритий ключі.
   - Що таке `KEY-ID` відбитка відкритого ключа `07A6 5898 2D3A F3DD 43E3 DA95 1F3F 3147 FA7F 54C7`?

     `FA7F 54C7`.
   - Який метод використовується для розподілу відкритих ключів на глобальному рівні?

     Сервери ключів.
3. Розмістіть наступні кроки в правильному порядку щодо відкликання закритого ключа:
   - Надайте відкликаний ключ своїм кореспондентам.
   - Створіть сертифікат відкликання.
   - Імпортуйте сертифікат відкликання до своєї зв’язки ключів.

     Правильний порядок:

Крок 1:	Створіть сертифікат про відкликання
Крок 2:	Імпортуйте сертифікат відкликання до своєї зв’язки ключів

Крок 3:	Надайте відкликаний ключ своїм кореспондентам

4. Стосовно шифрування файлів, що означає параметр `--armor` у команді `gpg --output encrypted-message --recipient carol --armor --encrypt unencrypted-message`?

   Він створює захищені вихідні дані у форматі ASCII, які дозволяють скопіювати отриманий існуючий зашифрований файл у електронний лист.

Відповіді до дослідницьких вправ

1. Більшість параметрів `gpg` мають як довгу, так і коротку версії. Доповніть таблицю відповідним коротким варіантом:

Довга версія	**Коротка версія**
`--armor`	`-a`
`--output`	`-o`
`--recipient`	`-r`
`--decrypt`	`-d`
`--encrypt`	`-e`
`--sign`	`-s`

2. Дайте відповідь на такі запитання щодо експорту ключів:
   - Яку команду ви використаєте, щоб експортувати всі свої відкриті ключі у файл під назвою `all.key`?

     `gpg --export --output all.key` або `gpg --export -o all.key`

   - Яку команду ви використали б, щоб експортувати всі свої приватні ключі у файл під назвою `all_private.key`?

     `gpg --export-secret-keys --output all_private.key` або `gpg --export-secret-keys -o all_private.key` (`--export-secret-keys` можна замінити на `--export-secret-subkeys` з дещо іншим результатом — перевірте `man pgp` для отримання додаткової інформації).

3. Який параметр `gpg` дозволяє виконувати більшість завдань, пов'язаних із керуванням ключами, показуючи вам меню?

   `--edit-key`

4. Який параметр `gpg` дозволяє зробити цифровий підпис відкритого тексту?

   `--clearsign`

Imprint

PDF згенерований: 2023-10-16

Навчальні матеріали LPI є ініціативою Linux Professional Institute (https://lpi.org). Навчальні матеріали та їх переклади можна знайти за адресою https://learning.lpi.org.

Для питань та коментарів щодо цього видання, а також щодо проекту в цілому, пишіть на електронну адресу: learning@lpi.org.