

Linux From Scratch
(Лінукс з нуля)
Версія 12.3
Опублікована 5 травня 2025

Створив: Gerard Beekmans
Головний редактор: Bruce Dubbs
Переклав: Петро Голуб

Linux From Scratch (Лінукс з нуля): Версія 12.3: Опубліковано 5 травня 2025

Автор: Gerard Beekmans та Головний редактор: Bruce Dubbs. Переклад українською: Петро Голуб

Copyright © 1999-2025 Gerard Beekmans

Copyright © 1999-2025, Gerard Beekmans

Усі права захищено.

Ця книга підпадає під ліцензію Creative Commons License.

Комп'ютерні інструкції можуть бути виокремлені з цієї книги на умовах ліцензії MIT License.

Linux® є зареєстрованою торговою маркою Лінуса Торвальдса.

ЗМІСТ

Передмова.....	ix
i. Вступне слово.....	ix
ii. Аудиторія.....	x
iii. LFS Цільові Архітектури.....	x
iv. Попередні вимоги.....	xi
v. LFS та Стандарти.....	xii
vi. Обґрунтування вибору пакетів у книзі.....	xiii
vii. Типографія.....	xx
viii. Структура.....	xxii
ix. Errata та Рекомендації з безпеки.....	xxiii
I. Вступ.....	1
1. Вступ.....	2
1.1. Як побудувати LFS систему.....	2
1.2. Що нового з останнього релізу.....	3
1.3. Журнал змін.....	5
1.4. Ресурси.....	10
1.5. Допомога.....	11
II. Підготовка до побудови.....	14
2. Підготовка хост-системи.....	15
2.1. Вступ.....	15
2.2. Вимоги до хост-системи.....	15
2.3. Етапи побудови LFS.....	18
2.4. Створення нового розділу.....	19
2.5. Створення файлової системи на розділі.....	22
2.6. Встановлення змінної \$LFS та значення umask.....	23
2.7. Монтування нового розділу.....	24
3. Пакети та патчі.....	26
3.1. Вступ.....	26
3.2. Всі пакети.....	27
3.3. Необхідні патчі.....	37
4. Фінальні приготування.....	38
4.1. Вступ.....	38

4.2. Створення обмеженої структури каталогів у файловій системі LFS.....	38
4.3. Додавання користувача LFS.....	39
4.4. Налаштування оточення.....	40
4.5. Про SBUs.....	43
4.6. Про набори тестів.....	44
III. Створення крос-компілятора LFS та тимчасових інструментів.....	46
Важливі попередні матеріали.....	47
Вступ.....	47
Технічні нотатки інструментарію.....	47
Загальні інструкції компіляції.....	53
5. Компіляція крос-інструментів.....	55
5.1. Вступ.....	55
5.2. Binutils-2.44 - Етап 1.....	56
5.3. GCC-14.2.0 - Етап 1.....	58
5.4. Linux-6.13.4 API Headers.....	61
5.5. Glibc-2.41.....	62
5.6. Libstdc++ із GCC-14.2.0.....	65
6. Крос-компіляція тимчасових інструментів.....	67
6.1. Вступ.....	67
6.2. M4-1.4.19.....	68
6.3. Ncurses-6.5.....	69
6.4. Bash-5.2.37.....	71
6.5. Coreutils-9.6.....	72
6.6. Diffutils-3.11.....	73
6.7. File-5.46.....	74
6.8. Findutils-4.10.0.....	75
6.9. Gawk-5.3.1.....	76
6.10. Grep-3.11.....	77
6.11. Gzip-1.13.....	78
6.12. Make-4.4.1.....	79
6.13. Patch-2.7.6.....	80
6.14. Sed-4.9.....	81
6.15. Tar-1.35.....	82
6.16. Xz-5.6.4.....	83

6.17. Binutils-2.44 - Pass 2.....	84
6.18. GCC-14.2.0 - Pass 2.....	85
7. Вступ до Chroot та створення додаткових тимчасових інструментів.....	88
7.1. Вступ.....	88
7.2. Зміна власника.....	88
7.3. Підготовка Virtual Kernel File Systems.....	88
7.4. Вхід до середовища Chroot.....	90
7.5. Створення директорій.....	91
7.6. Створення необхідних файлів і символічних посилань.....	92
7.7. Gettext-0.24.....	95
7.8. Bison-3.8.2.....	96
7.9. Perl-5.40.1.....	97
7.10. Python-3.13.2.....	98
7.11. Texinfo-7.2.....	99
7.12. Util-linux-2.40.4.....	100
7.13. Очищення та збереження тимчасової системи.....	102
IV. Збірка системи LFS.....	105
8. Встановлення базового системного програмного забезпечення.....	106
8.1. Вступ.....	106
8.2. Керування пакетами.....	107
8.3. Man-pages-6.12.....	112
8.4. Iana-Etc-20250123.....	113
8.5. Glibc-2.41.....	114
8.6. Zlib-1.3.1.....	123
8.7. Bzip2-1.0.8.....	124
8.8. Xz-5.6.4.....	126
8.9. Lz4-1.10.0.....	128
8.10. Zstd-1.5.7.....	129
8.11. File-5.46.....	130
8.12. Readline-8.2.13.....	131
8.13. M4-1.4.19.....	133
8.14. Bc-7.0.3.....	134
8.15. Flex-2.6.4.....	135
8.16. Tcl-8.6.16.....	136

8.17. Expect-5.45.4.....	138
8.18. DejaGNU-1.6.3.....	140
8.19. Pkgconf-2.3.0.....	141
8.20. Binutils-2.44.....	142
8.21. GMP-6.3.0.....	145
8.22. MPFR-4.2.1.....	147
8.23. MPC-1.3.1.....	148
8.24. Attr-2.5.2.....	149
8.25. Acl-2.3.2.....	150
8.26. Libcap-2.73.....	151
8.27. Libxcrypt-4.4.38.....	152
8.28. Shadow-4.17.3.....	154
8.29. GCC-14.2.0.....	159
8.30. Ncurses-6.5.....	165
8.31. Sed-4.9.....	168
8.32. Psmisc-23.7.....	169
8.33. Gettext-0.24.....	170
8.34. Bison-3.8.2.....	173
8.35. Grep-3.11.....	174
8.36. Bash-5.2.37.....	175
8.37. Libtool-2.5.4.....	177
8.38. GDBM-1.24.....	178
8.39. Gperf-3.1.....	179
8.40. Expat-2.6.4.....	180
8.41. Inetutils-2.6.....	181
8.42. Less-668.....	183
8.43. Perl-5.40.1.....	184
8.44. XML::Parser-2.47.....	187
8.45. Intltool-0.51.0.....	188
8.46. Autoconf-2.72.....	189
8.47. Automake-1.17.....	191
8.48. OpenSSL-3.4.1.....	192
8.49. Libelf із Elfutils-0.192.....	194
8.50. Libffi-3.4.7.....	195

8.51. Python-3.13.2.....	197
8.52. Flit-Core-3.11.0.....	200
8.53. Wheel-0.45.1.....	201
8.54. Setuptools-75.8.1.....	202
8.55. Ninja-1.12.1.....	203
8.56. Meson-1.7.0.....	204
8.57. Kmod-34.....	205
8.58. Coreutils-9.6.....	207
8.59. Check-0.15.2.....	214
8.60. Diffutils-3.11.....	215
8.61. Gawk-5.3.1.....	216
8.62. Findutils-4.10.0.....	218
8.63. Groff-1.23.0.....	219
8.64. GRUB-2.12.....	222
8.65. Gzip-1.13.....	225
8.66. IPRoute2-6.13.0.....	226
8.67. Kbd-2.7.1.....	228
8.68. Libpipeline-1.5.8.....	231
8.69. Make-4.4.1.....	232
8.70. Patch-2.7.6.....	233
8.71. Tar-1.35.....	234
8.72. Texinfo-7.2.....	235
8.73. Vim-9.1.1166.....	237
8.74. MarkupSafe-3.0.2.....	240
8.75. Jinja2-3.1.5.....	241
8.76. Udev із Systemd-257.3.....	242
8.77. Man-DB-2.13.0.....	246
8.78. Procps-ng-4.0.5.....	249
8.79. Util-linux-2.40.4.....	251
8.80. E2fsprogs-1.47.2.....	258
8.81. Sysklogd-2.7.0.....	262
8.82. SysVinit-3.14.....	264
8.82. Про символи налагодження.....	265
8.84. Видалення відлагоджувальної інформації.....	265

8.85. Очищення.....	268
9. Конфігурація системи.....	269
9.1. Вступ.....	269
9.2. LFS-Bootscripts-20240825.....	270
9.3. Огляд роботи з пристроями та модулями.....	272
9.4. Управління пристроями.....	276
9.5. Загальна конфігурація мережі.....	279
9.6. Використання та конфігурація скрипту завантаження System V.....	282
9.7. Налаштування локальних параметрів системи.....	293
9.8. Створення файлу /etc/inputrc.....	295
9.8. Створення файлу /etc/shells.....	296
10. Створення завантажувальної системи LFS.....	298
10.1. Вступ.....	298
10.2. Створення файлу /etc/fstab.....	298
10.3. Linux-6.13.4.....	300
10.4. Використання GRUB для налаштування процесу завантаження.....	307
11. Кінець.....	311
11.1. Кінець.....	311
11.2. Будьте враховані.....	311
11.3. Перезавантаження системи.....	312
11.4. Додаткові ресурси.....	313
11.5. Початок роботи після LFS.....	313

Передмова

Вступне слово

Моя подорож до вивчення і кращого розуміння Linux почалась ще у 1998. Я щойно встановив свій перший Linux дистрибутив і швидко став заінтересований всією концепцією і філософією, що стоїть за Linux.

Завжди існує безліч способів виконання одного завдання. Те саме можна сказати і про дистрибутиви Linux. Дуже багато їх існувало протягом років. Деякі з них існують і досі, деякі перетворилися на щось інше, а інші залишилися лише у наших спогадах. Усі вони працюють по-різному, щоб задоволити потреби своєї цільової аудиторії. Оскільки існує так багато різних способів досягнення однієї і тієї ж кінцевої мети, я почав усвідомлювати, що більше не муши обмежуватися жодним конкретним варіантом реалізації. До відкриття Linux ми просто мирилися з проблемами в інших операційних системах, оскільки не мали вибору. Це було те, що було, подобалося це вам чи ні. З Linux почала з'являтися концепція вибору. Якщо вам щось не подобалося, ви могли вільно, навіть заохочувалися, це змінити.

Я спробував кілька дистрибутивів і не міг визначитися з вибором. Вони були чудовими системами самі по собі. Це вже не було питанням правильного чи неправильного вибору. Це стало питанням особистого смаку. З цими усіма можливостями вибору, стало очевидним, що не буде одної системи, яка б була ідеальною для мене. З таким великим вибором стало очевидним, що не існує єдиної системи, яка б ідеально підходила саме мені. Тому я вирішив створити власну систему Linux, яка б повністю відповідала моїм особистим уподобанням.

Щоб створити справді власну систему, я вирішив компілювати все з вихідного коду, а не використовувати попередньо скомпільовані бінарні пакети. Ця «ідеальна» система Linux мала б сильні сторони різних систем без їхніх очевидних слабких сторін. Спочатку ця ідея здавалася досить складною. Проте я залишався переконаним, що таку систему можна створити.

Після вирішення таких проблем, як циклічні залежності та помилки компіляції, я нарешті створив власну систему Linux. Вона була повністю функціональною і цілком придатною для використання, як і будь-яка інша система Linux, що існувала на той час. Але це було моє власне творіння. Мені було дуже приємно самому зібрати таку систему. Єдине, що могло бути краще, — це самому створити кожну програму. Це було наступним найкращим варіантом.

Коли я поділився своїми цілями та досвідом з іншими членами спільноти Linux, стало очевидно, що ці ідеї викликають стійкий інтерес. Швидко стало зрозуміло, що такі спеціально розроблені системи Linux не тільки відповідають конкретним вимогам користувачів, але й служать ідеальною можливістю для програмістів та системних адміністраторів вдосконалити свої (існуючі) навички роботи з Linux. З цього розширеного інтересу і народився проект Linux From Scratch.

Ця книга «Linux From Scratch» є центральним ядром цього проекту. Вона містить інформацію та інструкції, необхідні для розробки та побудови власної системи. Хоча ця книга містить шаблон, який дозволить створити правильно працючу систему, ви можете вільно змінювати інструкції відповідно до своїх потреб, що є важливою частиною цього проекту. Ви зберігаєте контроль; ми лише допомагаємо вам розпочати власний шлях.

Я щиро сподіваюся, що ви отримаєте задоволення від роботи над власною системою Linux From Scratch і насолодитеся численними перевагами системи, яка є по-справжньому вашою.

--

Gerard Beekmans
gerard@linuxfromscratch.org

Аудиторія

Є багато причин, чому вам варто прочитати цю книгу. Одне з питань, яке задають багато людей, — «навіщо займатися ручним створенням системи Linux з нуля, якщо можна просто завантажити та встановити вже готову?»

Однією з важливих причин існування цього проекту є допомога вам у вивчені внутрішнього устрою системи Linux. Створення системи LFS допомагає продемонструвати, як працює Linux, як все взаємодіє і залежить одне від одного. Однією з найкращих переваг цього навчального досвіду є можливість налаштувати систему Linux відповідно до ваших індивідуальних потреб.

Ще одна ключова перевага LFS полягає в тому, що вона дає вам контроль над системою, не покладаючись на чужу реалізацію Linux. З LFS ви сидите за кермом. Ви диктуєте кожен аспект своєї системи.

LFS дозволяє створювати дуже компактні системи Linux. З іншими дистрибутивами ви часто змушені встановлювати велику кількість програм, які ви не використовуєте і не розумієте. Ці програми марнують ресурси. Ви можете заперечити, що з сучасними жорсткими дисками і процесорами марнування ресурсів більше не є проблемою. Однак іноді ви все одно обмежені розміром системи, якщо не брати до уваги інші фактори. Подумайте про завантажувальні CD, USB-накопичувачі та вбудовані системи. Це області, де LFS може бути корисним.

Ще однією перевагою спеціально розробленої системи Linux є безпека. Компілюючи всю систему з вихідного коду, ви отримуєте можливість перевіряти все і застосовувати всі необхідні патчі безпеки. Вам не доведеться чекати, поки хтось інший скомпілює бінарні пакети, що виправляють уразливість безпеки. Якщо ви не перевірите патч і не впровадите його самостійно, ви не матимете гарантії, що новий бінарний пакет був скомпільований правильно і належним чином виправляє проблему.

Метою Linux From Scratch є створення повноцінної та придатної для використання базової системи. Якщо ви не бажаєте створювати власну систему Linux з нуля, ви все одно можете скористатися інформацією, що міститься в цій книзі.

Існує надто багато вагомих причин для створення власної системи LFS, щоб перелічити їх усі тут. Зрештою, освіта є найважливішою причиною. Продовжуючи роботу з LFS, ви відкриєте для себе силу, яку можуть дати інформація та знання.

LFS Цільові Архітектури

Основними цільовими архітектурами LFS є процесори AMD/Intel x86 (32-бітні) та x86_64 (64-бітні). З іншого боку, інструкції в цій книзі також працюють, з деякими модифікаціями, з процесорами Power PC та ARM. Для побудови системи, що використовує один з цих альтернативних процесорів, основною передумовою, крім тих, що наведені на наступній сторінці, є наявність існуючої системи Linux, такої як попередня інсталляція

LFS, Ubuntu, Red Hat/Fedora, SuSE або інший дистрибутив, призначений для цієї архітектури. (Зверніть увагу, що 32-бітний дистрибутив можна інсталювати та використовувати як хост-систему на 64-бітному комп'ютері AMD/Intel.)

Виграш від побудови на 64-бітній системі, порівняно з 32-бітною системою, є мінімальним. Наприклад, у тестовій збірці LFS-9.1 на системі з процесором Core i7-4790, що використовує 4 ядра, були виміряні такі статистичні дані:

Architecture	Build Time	Build Size
32-bit	239.9 minutes	3.6 GB
64-bit	233.2 minutes	4.4 GB

Як бачите, на тому самому обладнанні 64-бітна збірка лише на 3% швидша (і на 22% більша) за 32-бітну збірку. Якщо ви плануєте використовувати LFS як сервер LAMP або брандмауер, 32-бітний процесор може бути достатнім. З іншого боку, декілька пакетів у BLFS тепер потребують більше 4 ГБ оперативної пам'яті для компіляції та/або роботи; якщо ви плануєте використовувати LFS як настільний комп'ютер, автори LFS рекомендують компілювати 64-бітну систему.

Стандартна 64-бітна збірка, що є результатом LFS, є «чистою» 64-бітною системою. Тобто вона підтримує лише 64-бітні виконувані файли. Створення «multi-lib» системи вимагає компіляції багатьох програм двічі: один раз для 32-бітної системи та один раз для 64-бітної системи. Це не підтримується безпосередньо в LFS, оскільки це суперечить освітній меті надання мінімальних інструкцій, необхідних для базової системи Linux. Деякі редактори LFS/BLFS підтримують мультибітовий форк LFS, доступний за адресою <https://www.linuxfromscratch.org/~thomas/multilib/index.html>. Але це вже тема для просунутих користувачів.

Попередні вимоги

Створення системи LFS — це не просте завдання. Воно вимагає певного рівня знань в області адміністрування систем Unix, щоб вирішувати проблеми і правильно виконувати перелічені команди. Зокрема, як абсолютний мінімум, ви вже повинні знати, як використовувати командний рядок (оболонку) для копіювання або переміщення файлів і каталогів, переліку вмісту каталогів і файлів, а також зміни поточного каталогу. Також очікується, що ви знаєте, як використовувати та встановлювати програмне забезпечення Linux.

Оскільки книга LFS передбачає принаймні цей базовий рівень знань, різні форуми підтримки LFS навряд чи нададуть вам значну допомогу в цих питаннях. Ви побачите, що ваші запитання щодо таких базових знань, ймовірно, залишаться без відповіді (або ж вас просто відправлять до переліку необхідних для LFS матеріалів для попереднього ознайомлення).

Перед тим, як створювати систему LFS, радимо прочитати ці статті:

- Software-Building-HOWTO <https://tldp.org/HOWTO/Software-Building-HOWTO.html>

Це вичерпний посібник із створення та встановлення «загальних» пакетів програмного забезпечення Unix під Linux. Хоча він був написаний деякий час тому, він все ще містить хороший огляд основних технік, що використовуються для створення та встановлення програмного забезпечення.

- Посібник для початківців з іnstalляції із вихідного коду <https://moi.vonos.net/linux/beginners-installing-from-source/>

Цей посібник містить хороший огляд основних навичок і технік, необхідних для створення програмного забезпечення з вихідного коду.

LFS та Стандарти

Структура LFS максимально відповідає стандартам Linux. Основні стандарти:

- POSIX.1-2008. [<https://pubs.opengroup.org/onlinepubs/9699919799/>]
- Стандарт ієрархії файлової системи (FHS) версія 3.0. [https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html]
- Базова стандартизація Linux (LSB) версія 5.0 (2015). [<https://refspecs.linuxfoundation.org/lsb.shtml>]

LSB має чотири окремі специфікації: Core, Desktop, Languages та Imaging. Деякі частини специфікацій Core та Desktop є специфічними для архітектури. Існують також дві пробні специфікації: Gtk3 та Graphics. LFS намагається відповідати специфікаціям LSB для архітектур IA32 (32-біт x86) або AMD64 (x86_64), про які йшлося в попередньому розділі.



Note

Багато людей не погоджуються з цими вимогами. Основною метою LSB є забезпечення можливості встановлення та запуску пропрієтарного програмного забезпечення на сумісній системі. Оскільки LFS базується на вихідному коді, користувач має повний контроль над тим, які пакети бажано встановити; ви можете вирішити не встановлювати деякі пакети, зазначені в LSB.

Хоча можна створити повну систему, яка пройде тести сертифікації LSB «з нуля», це неможливо зробити без багатьох додаткових пакетів, які виходять за рамки книги LFS. Інструкції з іnstalляції деяких з цих додаткових пакетів можна знайти в BLFS.

Пакети, що поставляються LFS для забезпечення відповідності вимогам LSB

LSB Core:	Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Gzip, M4, Man-DB, Procps, Psmisc, Sed, Shadow, SysVinit, Tar, Util-linux, Zlib
LSB Desktop:	None
LSB Languages:	Perl
LSB Imaging:	None
LSB Gtk3 and LSB Graphics (Trial Use):	None

Компоненти, які не постачаються або постачаються за бажанням LFS або BLFS, необхідні для задоволення вимог LSB

LSB Core:

install_initd, **libcrypt.so.1** (може бути наданий з опціональними інструкціями для пакета LFS Libcrypt), **libcurses.so.5** (може бути наданий з опціональними інструкціями для пакета LFS Ncurses), **libcursesw.so.5** (але **libcursesw.so.6** надається пакетом LFS Ncurses)

LSB Desktop:

libgdk-x11-2.0.so (але **libgdk-3.so** надається пакетом BLFS GTK+3), **libgtk-x11-2.0.so** (але **libgtk-3.so** та **libgtk-4.so** надаються пакетами BLFS GTK+3 та GTK-4), **libpng12.so** (але **libpng16.so** надається пакетом BLFS Libpng), **libQt*.so.4** (але **libQt6*.so.6** надаються пакетом BLFS Qt6), **libtiff.so.4** (але **libtiff.so.6** надається пакетом BLFS Libtiff)

LSB Languages:

/usr/bin/python (LSB вимагає Python2, але LFS і BLFS надають лише Python3)

LSB Imaging:

None

LSB Gtk3 and LSB Graphics (Trial Use):

libpng15.so (але **libpng16.so** надається пакетом BLFS Libpng)

Обґрунтування вибору пакетів у книзі

Метою LFS є створення повної та придатної для використання базової системи, що включає всі пакети, необхідні для її самовідтворення, та надання відносно мінімальної бази, на основі якої можна налаштовувати більш повну систему відповідно до вибору користувача. Це не означає, що LFS є найменшою можливою системою. До її складу входять кілька важливих пакетів, які, строго кажучи, не є обов'язковими. У наведеному нижче списку задокументовано причини включення кожного пакета до книги.

- **Acl**

Цей пакет містить утиліти для адміністрування списків контролю доступу, які використовуються для визначення деталізованих дискреційних прав доступу до файлів і каталогів.

- **Attr**

Цей пакет містить програми для управління розширеними атрибутами об'єктів файлової системи.

- **Autoconf**

Цей пакет містить програми для створення скриптів оболонки, які можуть автоматично налаштовувати вихідний код із шаблону розробника. Часто потрібно перекомпілювати пакет після оновлення процедури компіляції.

- **Automake**

Цей пакет містить програми для створення файлів Make з шаблону. Часто потрібно перекомпілювати пакет після оновлення процедури компіляції.

- Bash

Цей пакет задовольняє основну вимогу LSB щодо надання інтерфейсу Bourne Shell для системи. Він був обраний серед інших пакетів оболонок через його поширене використання та широкі можливості.

- Bc

Цей пакет надає мову обробки чисел довільної точності. Він задовольняє вимогу для побудови ядра Linux.

- Binutils

Цей пакет надає лінкер, асемблер та інші інструменти для обробки об'єктних файлів. Програми в цьому пакеті потрібні для компіляції більшості пакетів в системі LFS.

- Bison

Цей пакет містить GNU-версію yacc (Yet Another Compiler Compiler), необхідну для побудови декількох програм LFS.

- Bzip2

Цей пакет містить програми для стиснення та розпакування файлів. Він необхідний для розпакування багатьох пакетів LFS.

- Check

Цей пакет надає тестовий інструментарій для інших програм.

- Coreutils

Цей пакет містить низку необхідних програм для перегляду та обробки файлів і каталогів. Ці програми потрібні для управління файлами з командного рядка і необхідні для процедур встановлення кожного пакета в LFS.

- DejaGNU

Цей пакет надає інфраструктуру для тестування інших програм.

- Diffutils

Цей пакет містить програми, які показують відмінності між файлами або каталогами. Ці програми можуть бути використані для створення патчів, а також використовуються в процедурах побудови багатьох пакетів.

- E2fsprogs

Цей пакет містить утиліти для роботи з файловими системами ext2, ext3 та ext4. Це найпоширеніші та найбільш ретельно протестовані файлові системи, які підтримує Linux.

- Expat

Цей пакет містить відносно невелику бібліотеку для аналізу XML. Вона потрібна для модуля Perl XML::Parser.

- Expect

Цей пакет містить програму для виконання скриптових діалогів з іншими інтерактивними програмами. Він зазвичай використовується для тестування інших пакетів.

- File

Цей пакет містить утиліту для визначення типу певного файлу або файлів. Декілька пакетів потребують її у своїх скриптах побудови.

- Findutils

Цей пакет надає програми для пошуку файлів у файловій системі. Він використовується у скриптах побудови багатьох пакетів.

- Flex

Цей пакет містить утиліту для генерації програм, що розпізнають шаблони у тексті. Це GNU-версія програми lex (лексичний аналізатор). Вона необхідна для побудови декількох пакетів LFS.

- Gawk

Цей пакет містить програми для обробки текстових файлів. Це GNU-версія awk (Aho-Weinberg- Kernighan). Він використовується в багатьох інших скриптах збірки пакетів.

- GCC

Це колекція компіляторів Gnu. Вона містить компілятори C і C++, а також кілька інших, які не збираються в LFS.

- GDBM

Цей пакет містить бібліотеку GNU Database Manager. Він використовується одним іншим пакетом LFS, Man-DB.

- Gettext

Цей пакет надає утиліти та бібліотеки для інтернаціоналізації та локалізації багатьох пакетів.

- Glibc

Цей пакет містить основну бібліотеку С. Без неї програми Linux не працюватимуть.

- GMP

Цей пакет містить математичні бібліотеки, що надають корисні функції для арифметики довільної точності. Він необхідний для компіляції GCC.

- Gperf

Цей пакет створює програму, яка генерує ідеальну хеш-функцію з набору ключів. Він необхідний для Udev.

- Grep

Цей пакет містить програми для пошуку у файлах. Ці програми використовуються більшістю скриптів компіляції пакетів.

- Groff

Цей пакет містить програми для обробки та форматування тексту. Однією з важливих функцій цих програм є форматування сторінок man.

- GRUB

Це Grand Unified Boot Loader (Великий уніфікований завантажувач). Це найгнучкіший з декількох доступних завантажувачів.

- Gzip

Цей пакет містить програми для стиснення та розпакування файлів. Він необхідний для розпакування багатьох пакетів в LFS.

- Iana-etc

Цей пакет надає дані для мережевих служб та протоколів. Він необхідний для забезпечення належних мережевих можливостей.

- Inetutils

Цей пакет містить програми для базового адміністрування мережі.

- Intltool

Цей пакет містить інструменти для вилучення перекладних рядків з вихідних файлів.

- IProute2

Цей пакет містить програми для базової та розширеної роботи з мережами IPv4 та IPv6. Він був обраний замість іншого поширеного пакету мережевих інструментів (net-tools) через його можливості роботи з IPv6.

- Kbd

Цей пакет створює файли таблиць клавіш, утиліти для клавіатур, що не є американськими, та низку шрифтів для консолі.

- Less

Цей пакет містить дуже зручний переглядач текстових файлів, який дозволяє прокручувати файл вгору або вниз під час перегляду. Багато пакетів використовують його для сторінкування виводу.

- **Libcap**

Цей пакет реалізує інтерфейси користувачького простору для можливостей POSIX 1003.1e, доступних у ядрах Linux.

- **Libelf**

Проект elfutils надає бібліотеки та інструменти для файлів ELF та даних DWARF. Більшість утиліт у цьому пакеті доступні в інших пакетах, але бібліотека потрібна для побудови ядра Linux з використанням стандартної (і найбільш ефективної) конфігурації.

- **Libffi**

Цей пакет реалізує портативний високорівневий інтерфейс програмування для різних конвенцій виклику. Деякі програми можуть не знати під час компіляції, які аргументи мають бути передані функції. Наприклад, інтерпретатор може отримати інформацію під час виконання про кількість і типи аргументів, що використовуються для виклику певної функції. Libffi може використовуватися в таких програмах для забезпечення зв'язку між програмою-інтерпретатором і скомпільованим кодом.

- **Libpipeline**

Пакет Libpipeline надає бібліотеку для гнучкого та зручного керування конвеєрами підпроцесів. Він необхідний для пакета Man-DB.

- **Libtool**

Цей пакет містить скрипт підтримки загальних бібліотек GNU. Він об'єднує складність використання спільнотних бібліотек у послідовний, портативний інтерфейс. Він необхідний для наборів тестів в інших пакетах LFS.

- **Libxcrypt**

Цей пакет надає бібліотеку libcrypt, необхідну для різних пакетів (зокрема, Shadow) для хешування паролів. Він замінює застарілу реалізацію libcrypt в Glibc.

- **Linux Kernel**

Цей пакет - це Операційна система. Саме він є Linux в середовищі GNU/Linux.

- **M4**

Цей пакет надає загальний текстовий макропроцесор, корисний як інструмент побудови для інших програм.

- **Make**

Цей пакет містить програму для керування побудовою пакетів. Він необхідний майже для кожного пакета в LFS.

- Man-DB

Цей пакет містить програми для пошуку та перегляду сторінок man. Він був обраний замість пакета man через його кращі можливості інтернаціоналізації. Він надає програму man.

- Man-pages

Цей пакет надає фактичний вміст основних сторінок man Linux.

- Meson

Цей пакет надає програмний інструмент для автоматизації побудови програмного забезпечення. Головною метою Meson є мінімізація часу, який розробники програмного забезпечення витрачають на налаштування системи побудови. Він необхідний для побудови Systemd, а також багатьох пакетів BLFS.

- MPC

Цей пакет надає арифметичні функції для комплексних чисел. Він необхідний для GCC.

- MPFR

Цей пакет містить функції для арифметики з багаторазовою точністю. Він необхідний для GCC.

- Ninja

Цей пакет надає невелику систему побудови, орієнтовану на швидкість. Він розроблений для того, щоб його вхідні файли генерувалися системою побудови вищого рівня і щоб побудова виконувалася якомога швидше. Цей пакет необхідний для Meson.

- Ncurses

Цей пакет містить бібліотеки для незалежної від терміналу обробки символічних екранів. Він часто використовується для забезпечення керування курсором для системи меню. Він необхідний для ряду пакетів в LFS.

- Openssl

Цей пакет надає інструменти управління та бібліотеки, пов'язані з криптографією. Вони надають криптографічні функції іншим пакетам, включаючи ядро Linux.

- Patch

Цей пакет містить програму для модифікації або створення файлів шляхом застосування файлу патча, який зазвичай створюється програмою diff. Він необхідний для процедури побудови декількох пакетів LFS.

- Perl

Цей пакет є інтерпретатором мови виконання PERL. Він необхідний для інсталяції та набору тестів декількох пакетів LFS.

- Pkgconf

Цей пакет містить програму, яка допомагає налаштувати пропори компілятора та лінкера для бібліотек розробки. Програма може бути використана як заміна pkg-config, яка потрібна для системи побудови багатьох пакетів. Вона підтримується більш активно і працює трохи швидше, ніж оригінальний пакет Pkg-config.

- Procps-NG

Цей пакет містить програми для моніторингу процесів. Ці програми корисні для системного адміністрування, а також використовуються LFS Bootscripts.

- Psmisc

Цей пакет містить програми для відображення інформації про запущені процеси. Ці програми корисні для системного адміністрування.

- Python 3

Цей пакет надає інтерпретовану мову, філософія розробки якої наголошує на читабельності коду.

- Readline

Цей пакунок містить бібліотеки, які дозволяють редагувати текст у командному рядку та зберігати історію введених команд. Цю бібліотеку використовує оболонка Bash.

- Sed

Цей пакет дозволяє редагувати текст без відкриття його в текстовому редакторі. Він також необхідний для багатьох скриптів конфігурації пакетів LFS.

- Shadow

Цей пакет містить програми для безпечної управління паролями.

- Sysklogd

Цей пакет містить програми для реєстрації системних повідомлень, таких як ті, що видаються ядром або демонами процесами при виникненні незвичайних подій.

- SysVinit

Цей пакет містить програму init, яка є батьківським процесом для всіх інших процесів у працюючій системі Linux.

- Udev

Цей пакет є менеджером пристройів. Він динамічно контролює власність, дозволи, імена та символічні посилання вузлів пристройів у каталозі /dev, коли пристрой додається до системи або видаляється з неї.

- Tar

Цей пакет надає можливості архівування та розпакування практично всіх пакетів, що використовуються в LFS.

- **Tcl**

Цей пакет містить Toll Command Language, яка використовується в багатьох наборах тестів.

- **Texinfo**

Цей пакет містить програми для читання, запису та перетворення інформаційних сторінок. Він використовується в процедурах встановлення багатьох пакетів LFS.

- **Util-linux**

Цей пакет містить різні утиліти. Серед них є утиліти для роботи з файловими системами, консолями, розділами та повідомленнями.

- **Vim**

Цей пакет надає редактор. Він був обраний через його сумісність з класичним редактором vi та величезну кількість потужних можливостей. Редактор є дуже особистим вибором для багатьох користувачів. Може бути замінений на будь-який інший редактор за бажанням, наприклад nano.

- **Wheel**

Цей пакет надає модуль Python, який є еталонною реалізацією стандарту пакування Python wheel.

- **XML::Parser**

Цей пакет є модулем Perl, який взаємодіє з Expat.

- **XZ Utils**

Цей пакет містить програми для стиснення та розпакування файлів. Він забезпечує найвищий рівень стиснення, який зазвичай доступний, і є корисним для розпакування пакетів у форматі XZ або LZMA.

- **Zlib**

Цей пакет містить процедури стиснення та розпакування, які використовуються деякими програмами.

- **Zstd**

Цей пакет містить процедури стиснення та розпакування, які використовуються деякими програмами. Він забезпечує високий коефіцієнт стиснення та дуже широкий діапазон співвідношень між стисненням і швидкістю.

Типографія

Для зручності сприйняття в цій книзі використовуються деякі типографські умовні позначення. Цей розділ містить кілька прикладів типографічного форматування, яке використовується в книзі «Linux From Scratch».

```
./configure --prefix=/usr
```

Ця форма тексту призначена для введення його точно так, як він вказаний, якщо інше не зазначається в сусідньому тексті. Вона також використовується в розділах пояснень для ідентифікації команди, на яку є посилання.

У деяких випадках логічний рядок розширяється до двох або більше фізичних рядків із символом зворотного слеша в кінці рядка.

```
CC="gcc -B/usr/bin/" ..../binutils-2.18/configure \
--prefix=/tools --disable-nls --disable-werror
```

Зверніть увагу, що за зворотним слешем повинен слідувати негайний перехід на наступний рядок. Інші символи, такі як пробіли або табуляція, призведуть до неправильних результатів.

```
install-info: unknown option '--dir-file=/mnt/lfs/usr/info/dir'
```

Ця форма тексту відображає вихідні дані на екрані, зазвичай як результат виконаних команд. Цей формат також використовується для відображення імен файлів, таких як /etc/ld.so.conf.



Note

Будь ласка, налаштуйте свій браузер для відображення тексту фіксованої ширини з хорошим монопросторовим шрифтом font-size="9ptd, за допомогою якого ви зможете чітко розрізнати символи 111 або 00.

Курсив

Ця форма тексту використовується в книзі для кількох цілей. Її основна мета — виділити важливі моменти або елементи.

<https://www.linuxfromscratch.org/>

Цей формат використовується для гіперпосилань як всередині спільноти LFS, так і на зовнішні сторінки. Він включає HOWTO, місця завантаження та веб- сайти.

```
cat > $LFS/etc/group << "EOF"
root:x:0:
bin:x:1:
.....
EOF
```

Цей формат використовується при створенні файлів конфігурації. Перша команда вказує системі створити файл \$LFS/etc/group з усього, що введено в наступних рядках, доки не буде знайдено послідовність End Of File (EOF). Тому вся ця секція, як правило, вводиться так, як вона виглядає.

<*REPLACED TEXT*>

Цей формат використовується для інкапсуляції тексту, який не повинен вводитися так, як він є, або копіюватись та вставлятись.

[OPTIONAL TEXT]

Цей формат використовується для інкапсуляції тексту, який є необов'язковим.

passwd(5)

Цей формат використовується для посилання на конкретну сторінку довідника (*man*). Число в дужках вказує на конкретний розділ в довіднику. Наприклад, **passwd** має дві сторінки довідника. Згідно з інструкціями з інсталяції LFS, ці дві сторінки довідника знаходяться в */usr/share/man/man1/passwd.1* та */usr/share/man/man5/passwd.5*. Коли в книзі використовується *passwd(5)*, це конкретно посилання на */usr/share/man/man5/passwd.5*. **man passwd** виведе першу сторінку *man*, яку знайде, що відповідає «*passwd*», а це буде */usr/share/man/man1/passwd.1*. Для цього прикладу вам потрібно буде запустити **man 5 passwd**, щоб прочитати вказану сторінку. Зверніть увагу, що більшість сторінок *man* не мають дублікатів імен сторінок у різних розділах. Тому, як правило, достатньо ввести **man <назва програми>**. У книзі LFS ці посилання на сторінки *man* також є гіперпосиланнями, тому натискання на таке посилання відкриє сторінку *man*, відтворену в HTML з сторінок посібника Arch Linux.

Структура

Ця книга поділена на такі частини.

Частина I – Вступ

У частині I наведено кілька важливих зауважень щодо встановлення LFS. У цьому розділі також наведено метаінформацію про книгу.

Частина II – Підготовка до побудови

У частині II описано, як підготуватися до процесу побудови — створення розділу, завантаження пакетів та компіляції тимчасових інструментів.

Частина III – Створення крос-компілятора LFS та тимчасових інструментів

У частині III наведено інструкції щодо створення інструментів, необхідних для побудови остаточної системи LFS.

Частина IV – Побудова системи LFS

Частина IV супроводжує читача через процес побудови системи LFS — компіляцію та встановлення всіх пакетів один за одним, налаштування скриптів завантаження та встановлення ядра. Отримана система Linux є основою, на якій можна будувати інше програмне забезпечення для розширення системи за бажанням. В кінці цієї книги є простий у використанні довідник, що містить перелік всіх програм, бібліотек та важливих файлів, які були встановлені.

Частина V – Додатки

Частина V містить інформацію про саму книгу, включаючи абревіатури та терміни, подяки, залежності пакетів, перелік завантажувальних скриптів LFS, ліцензії на розповсюдження книги та вичерпний покажчик пакетів, програм, бібліотек та скриптів.

Errata та Рекомендації з безпеки

Програмне забезпечення, яке використовується для створення системи LFS, постійно оновлюється та вдосконалюється. Попередження про безпеку та виправлення помилок можуть з'явитися після виходу книги LFS. Щоб перевірити, чи потребують версії пакетів або інструкції в цьому випуску LFS будь-яких змін — для усунення вразливостей безпеки або виправлення інших помилок — відвідайте <https://www.linuxfromscratch.org/lfs/errata/12.3/> перед тим, як продовжувати збірку. Ви повинні відзначити всі показані зміни та застосувати їх до відповідних розділів книги під час збірки системи LFS.

Крім того, редактори Linux From Scratch ведуть список вразливостей безпеки, виявлених після виходу книги. Щоб ознайомитися з цим списком, відвідайте <https://www.linuxfromscratch.org/lfs/advisories/> перед тим, як приступити до збірки. Ви повинні застосувати зміни, запропоновані в рекомендаціях, до відповідних розділів книги під час збірки системи LFS. А якщо ви будете використовувати систему LFS як справжню настільну або серверну систему, вам слід продовжувати користуватися цими рекомендаціями та виправляти будь-які вразливості безпеки, навіть після повного створення системи LFS.

Частина I. Вступ

Глава 1. Вступ

1.1. Як побудувати LFS систему

Система LFS будеться з використанням вже встановленого дистрибутива Linux (такого як Debian, OpenMandriva, Fedora або openSUSE). Ця існуюча система Linux (хост) буде використовуватися як відправна точка для надання необхідних програм, включаючи компілятор, лінкер і оболонку, для побудови нової системи. Виберіть опцію «розробка» під час встановлення дистрибутива, щоб включити ці інструменти.



Note

Існує багато способів встановлення дистрибутива Linux, і стандартні налаштування зазвичай не є оптимальними для побудови системи LFS. Рекомендації щодо налаштування комерційного дистрибутива див. за адресою: <https://www.linuxfromscratch.org/hints/downloads/files/partitioning-for-lfs.txt>.

Як альтернативу встановленню окремого дистрибутиву на вашому комп'ютері, ви можете скористатися LiveCD з комерційним дистрибутивом.

У розділі 2 цієї книги описано, як створити новий розділ і файлову систему для Linux, де буде скомпільована і встановлена нова система LFS. У розділі 3 пояснено, які пакети і патчі необхідно завантажити для побудови системи LFS, і як зберігати їх у новій файловій системі. У розділі 4 обговорюється налаштування відповідного робочого середовища. Будь ласка, уважно прочитайте розділ 4, оскільки в ньому пояснюються кілька важливих питань, про які ви повинні знати, перш ніж почати роботу з розділом 5 і далі.

У розділі 5 пояснюється встановлення початкового набору інструментів (binutils, gcc та glibc) з використанням методів крос-компіляції для ізоляції нових інструментів від хост-системи.

У розділі 6 показано, як крос-компілювати основні утиліти за допомогою щойно створеного крос-інструментарію.

У розділі 7 ми переходимо до середовища «chroot», де використовуємо нові інструменти для створення всіх інших інструментів, необхідних для створення системи LFS.

Ці зусилля з ізоляції нової системи від хост-дистрибутива можуть здатися надмірними. Повне технічне пояснення того, чому це робиться, наведено в Технічних примітках до інструментарію.

У розділі 8 створюється повноцінна система LFS. Ще однією перевагою середовища chroot є те, що воно дозволяє продовжувати використовувати хост-систему під час створення LFS. Поки ви чекаєте на завершення компіляції пакетів, ви можете продовжувати використовувати свій комп'ютер як зазвичай.

Щоб завершити встановлення, у розділі 9 налаштовується базова конфігурація системи, а ядро та завантажувач створюються у розділі 10. Розділ 11 містить інформацію про продовження роботи з LFS після прочитання цієї книги. Після виконання кроків, описаних у цьому розділі, комп'ютер готовий до завантаження нової системи LFS. Це короткий опис процесу. Детальна інформація про кожен крок наведена в наступних розділах. Пункти, які зараз здаються складними, будуть пояснені, і все стане на свої місця, коли ви почнете свою пригоду з LFS.

1.2. Що нового з останнього релізу

Ось список пакетів, оновлених з моменту виходу попередньої версії LFS.

Оновлено до:

- Bash-5.2.37
- Bc-7.0.3
- Binutils-2.44
- Coreutils-9.6
- Diffutils-3.11
- E2fsprogs-1.47.2
- Expat-2.6.4
- File-5.46
- Flit-core-3.11.0
- Gawk-5.3.1
- Gettext-0.24
- Glibc-2.41
- Iana-Etc-20250123
- Inetutils-2.6
- IPRoute2-6.13.0
- Jinja2-3.1.5
- Kbd-2.7.1
- Kmod-34
- Less-668
- Libcap-2.73
- Libelf from Elfutils-0.192
- Libffi-3.4.7
- Libpipeline-1.5.8
- Libtool-2.5.4

- Libxcrypt-4.4.38
- Linux-6.13.4
- Man-DB-2.13.0
- Man-pages-6.12
- MarkupSafe-3.0.2
- Meson-1.7.0
- OpenSSL-3.4.1
- Perl-5.40.1
- Procps-ng-4.0.5
- Python-3.13.2
- Setuptools-75.8.1
- Shadow-4.17.3
- Sysklogd-2.7.0
- Systemd-257.3
- SysVinit-3.14Жу
- Tcl-8.6.16
- Texinfo-7.2
- Tzdata-2025a
- Udev from Systemd-257.3
- Util-linux-2.40.4
- Vim-9.1.1166
- Wheel-0.45.1
- Xz-5.6.4
- Zstd-1.5.7

Додано:

-

Видалено:

1.3. Журнал змін

Це версія 12.3 книги «Linux From Scratch» від 5 березня 2025 року. Якщо цій книзі більше шести місяців, ймовірно, вже доступна новіша і краща версія. Щоб дізнатися про це, перевірте один із дзеркал за адресою <https://www.linuxfromscratch.org/mirrors.html>.

Нижче наведено перелік змін, внесених з моменту попереднього випуску книги.

Записи в журналі змін:

- 2025-03-05
 - [bdubbs] - LFS-12.3 released.
- 2025-03-02
 - [bdubbs] - Update to vim-9.1.1166 (Security Update). Fixes #5666.
- 2025-02-27
 - [bdubbs] - Update to zstd-1.5.7. Fixes #5652.
 - [bdubbs] - Update to systemd-257.3. Fixes #5612.
 - [bdubbs] - Update to shadow-4.17.3. Fixes #5660.
 - [bdubbs] - Update to setuptools-75.8.1. Fixes #5662.
 - [bdubbs] - Update to linux-6.13.4. Fixes #5647.
 - [bdubbs] - Update to kmod-34. Fixes #5657.
 - [bdubbs] - Update to inetutils-2.6. Fixes #5656.
 - [bdubbs] - Update to gettext-0.24. Fixes #5661.
 - [bdubbs] - Update to flit_core-3.11.0. Fixes #5654.
- 2025-02-24
 - [bdubbs] - Update to man-pages-6.12. Fixes #5658.
- 2025-02-19
 - [xry111] - Update to vim-9.1.1122 (Security Update). Addresses #4500.
 - [xry111] - Update to man-pages-6.11. Fixes #5646.
- 2025-02-13

- [bdubbs] - Update to vim-9.1.1106. Addresses #4500.
- [bdubbs] - Update to diffutils-3.11. Fixes #5639.
- [bdubbs] - Update to libffi-3.4.7. Fixes #5642.
- [bdubbs] - Update to linux-6.13.2. Fixes #5643.
- [bdubbs] - Update to Python3-3.13.2. Fixes #5640.
- [bdubbs] - Update to sysvinit-3.14. Fixes #5641.
- 2025-02-02
 - [bdubbs] - Update to vim-9.1.1071. Addresses #4500.
 - [bdubbs] - Update to iana-etc-20250123. Addresses #5006.
 - [bdubbs] - Update to binutils-2.44.0. Fixes #5634.
 - [bdubbs] - Update to coreutils-9.6. Fixes #5628.
 - [bdubbs] - Update to e2fsprogs-1.47.2. Fixes #5637.
 - [bdubbs] - Update to glibc-2.41. Fixes #5638.
 - [bdubbs] - Update to iproute2-6.13.0. Fixes #5631.
 - [bdubbs] - Update to libxcrypt-4.4.38. Fixes #5626.
 - [bdubbs] - Update to linux-6.13.1. Fixes #5629.
 - [bdubbs] - Update to man-pages-6.10. Fixes #5632.
 - [bdubbs] - Update to meson-1.7.0. Fixes #5636.
 - [bdubbs] - Update to perl-5.40.1. Fixes #5630.
 - [bdubbs] - Update to tcl8.6.16. Fixes #5635.
 - [bdubbs] - Update to tzdata2025a. Fixes #5627.
 - [bdubbs] - Update to xz-5.6.4. Fixes #5633.
- 2025-01-15
 - [bdubbs] - Update to vim-9.1.1016. Addresses #4500.
 - [bdubbs] - Update to iana-etc-20250108. Addresses #5006.
 - [bdubbs] - Update to util-linux-2.40.4. Fixes #5624.
 - [bdubbs] - Update to sysvinit-3.13. Fixes #5621.

- [bdubbs] - Update to sysklogd-2.7.0. Fixes #5623.
- [bdubbs] - Update to shadow-4.17.2. Fixes #5625.
- [bdubbs] - Update to setuptools-75.8.0. Fixes #5622.
- [bdubbs] - Update to linux-6.12.9. Fixes #5620.
- [bdubbs] - Update to gettext-0.23.1. Fixes #5619.
- 2025-01-01
 - [renodr] - Update to libxcrypt-4.4.37. Fixes #5618.
 - [bdubbs] - Update to iana-etc-20241220. Addresses #5006.
 - [bdubbs] - Update to texinfo-7.2. Fixes #5616.
 - [bdubbs] - Update to sysvinit-3.12. Fixes #5615.
 - [bdubbs] - Update to shadow-4.17.1. Fixes #5617.
 - [bdubbs] - Update to procps-ng-4.0.5. Fixes #5611.
 - [bdubbs] - Update to meson-1.6.1. Fixes #5610.
 - [bdubbs] - Update to linux-6.12.7. Fixes #5613.
 - [bdubbs] - Update to kbd-2.7.1. Fixes #5608.
 - [bdubbs] - Update to jinja2-3.1.5 (Security Update). Fixes #5614.
- 2024-12-15
 - [bdubbs] - Update to vim-9.1.0927. Addresses #4500.
 - [bdubbs] - Update to iana-etc-20241206. Addresses #5006.
 - [bdubbs] - Update to systemd-257. Fixes #5559.
 - [bdubbs] - Update to Python-3.13.1 (Security Update). Fixes #5605.
 - [bdubbs] - Update to libcap-2.73. Fixes #5604.
 - [bdubbs] - Update to linux-6.12.5. Fixes #5607.
 - [bdubbs] - Update to kbd-2.7. Fixes #5608.
 - [bdubbs] - Update to gettext-0.23. Fixes #5603.
- 2024-12-01
 - [bdubbs] - Update to iana-etc-20241122. Addresses #5006.

- [bdubbs] - Update to file-5.46. Fixes #5601.
- [bdubbs] - Update to iproute2-6.12.0. Fixes #5597.
- [bdubbs] - Update to libtool-2.5.4. Fixes #5598.
- [bdubbs] - Update to linux-6.12.1. Fixes #5586.
- [bdubbs] - Update to setuptools-75.6.0 (Python Module). Fixes #5599.
- [bdubbs] - Update to wheel-0.45.1 (Python Module). Fixes #5600.
- 2024-11-15
 - [bdubbs] - Update to vim-9.1.0866. Addresses #4500.
 - [bdubbs] - Update to iana-etc-20241024. Addresses #5006.
 - [bdubbs] - Update to wheel-0.45.0 (Python Module). Fixes #5593.
 - [bdubbs] - Update to setuptools-75.5.0 (Python Module). Fixes #5595.
 - [bdubbs] - Update to linux-6.11.8. Fixes #5582.
 - [bdubbs] - Update to libcap-2.72. Fixes #5594.
- 2024-11-08
 - [bdubbs] - Added binutils-2.43.1-upstream_fix-1.patch. Fixes #5591.
 - [bdubbs] - Update to flit_core-3.10.1. Fixes #5589.
 - [bdubbs] - Update to expat-2.6.4. Fixes #5590.
- 2024-10-25
 - [bdubbs] - Update to linux-6.11.6. Fixes #5588.
 - [bdubbs] - Update to libcap-2.71. Fixes #5584.
 - [bdubbs] - Update to setuptools-75.3.0. Fixes #5585.
 - [bdubbs] - Update to flit_core-3.10.0. Fixes #5587.
- 2024-10-25
 - [bdubbs] - Update to iana-etc-20241015. Addresses #5006.
 - [bdubbs] - Update to vim-9.1.0813. Addresses #4500.
 - [bdubbs] - Update to xz-5.6.3. Fixes #5572.
 - [bdubbs] - Update to sysvinit-3.11. Fixes #5581.

- [bdubbs] - Update to setuptools-75.2.0. Fixes #5577.
- [bdubbs] - Update to Python3-3.13.0. Fixes #5575.
- [bdubbs] - Update to openssl-3.4.0. Fixes #5582.
- [bdubbs] - Update to meson-1.6.0. Fixes #5580.
- [bdubbs] - Update to markupsafe-3.0.2. Fixes #5576.
- [bdubbs] - Update to linux-6.11.5. Fixes #5574.
- [bdubbs] - Update to less-668. Fixes #5578.
- [bdubbs] - Update to elfutils-0.192. Fixes #5579.
- 2024-10-03
 - [bdubbs] - Revert back to tcl8.6.15.
- 2024-10-01
 - [bdubbs] - Update to Python3-3.12.7. Fixes #5571.
 - [bdubbs] - Update to tcl9.0.0. Fixes #5570.
 - [bdubbs] - Update to linux-6.11.1. Fixes #5556.
 - [bdubbs] - Update to libtool-2.5.3. Fixes #5569.
 - [bdubbs] - Update to iproute2-6.11.0. Fixes #5561.
 - [bdubbs] - Update to bash-5.2.37. Fixes #5567.
 - [bdubbs] - Update to bc-7.0.3. Fixes #5568.
- 2024-09-20
 - [bdubbs] - Update to vim-9.1.0738. Addresses #4500.
 - [bdubbs] - Update to texinfo-7.1.1. Fixes #5558.
 - [bdubbs] - Update to tcl8.6.15. Fixes #5562.
 - [bdubbs] - Update to sysklogd-2.6.2. Fixes #5557.
 - [bdubbs] - Update to setuptools-75.1.0. Fixes #5560.
 - [bdubbs] - Update to meson-1.5.2. Fixes #5566.
 - [bdubbs] - Update to iana-etc-20240912. Addresses #5006.
 - [bdubbs] - Update to gawk-5.3.1. Fixes #5564.

- [bdubbs] - Update to bc-7.0.2. Fixes #5563.
- 2024-09-07
 - [bdubbs] - Update to tzdata-2024b. Fixes #5554.
 - [bdubbs] - Update to systemd-256.5. Fixes #5551.
 - [bdubbs] - Update to setuptools-74.1.2. Fixes #5546.
 - [bdubbs] - Update to python3-3.12.6. Fixes #5555.
 - [bdubbs] - Update to openssl-3.3.2. Fixes #5552.
 - [bdubbs] - Update to man-db-2.13.0. Fixes #5550.
 - [bdubbs] - Update to linux-6.10.8. Fixes #5545.
 - [bdubbs] - Update to libpipeline-1.5.8. Fixes #5548.
 - [bdubbs] - Update to expat-2.6.3. Fixes #5553.
 - [bdubbs] - Update to bc-7.0.1. Fixes #5547.
- 2024-09-01
 - [bdubbs] - LFS-12.2 released.

1.4. Ресурси

1.4.1. FAQ

Якщо під час побудови системи LFS ви зіткнулися з помилками, у вас виникли питання або ви вважаєте, що в книзі є друкарська помилка, спочатку зверніться до списку поширених питань (FAQ), який знаходитьться за адресою <https://www.linuxfromscratch.org/faq/>.

1.4.2. Списки розсилки

Сервер [linuxfromscratch.org](https://www.linuxfromscratch.org) містить низку списків розсилки, які використовуються для розвитку проекту LFS. Ці списки включають, серед інших, основні списки розвитку та підтримки. Якщо ви не можете знайти відповідь на своє питання на сторінці FAQ, наступним кроком буде пошук у списках розсилки на <https://www.linuxfromscratch.org/search.html>.

Інформацію про різні списки, як підписатися, місця архівування та додаткову інформацію можна знайти на <https://www.linuxfromscratch.org/mail.html>.

1.4.3. IRC

Декілька членів спільноти LFS пропонують допомогу через Internet Relay Chat (IRC). Перш ніж скористатися цією підтримкою, переконайтесь, що відповідь на ваше запитання вже не міститься у FAQ LFS або архівах

списку розсилки. Ви можете знайти мережу IRC за адресою <irc.libera.chat>. Канал підтримки має назву #lfs-support.

1.4.4. Сайти-дзеркала

Проект LFS має низку сайтів-дзеркал по всьому світу, щоб зробити доступ до веб-сайту та завантаження необхідних пакетів більш зручним. Будь ласка, відвідайте веб-сайт LFS за адресою <https://www.linuxfromscratch.org/mirrors.html>, щоб переглянути список поточних сайтів-дзеркал.

1.4.5. Контактна інформація

Будь ласка, надсилайте всі свої запитання та коментарі на одну з поштових розсилок LFS (див. Вище).

1.5. Допомога



Note

Якщо ви зіткнулися з проблемою під час створення одного пакета за інструкцією LFS, ми настійно не рекомендуємо публікувати цю проблему безпосередньо в каналі підтримки upstream, не обговоривши її попередньо в каналі підтримки LFS, зазначеному в розділі 1.4 «Ресурси». Такі дії часто є неефективними, оскільки розробники рідко знайомі з процедурою створення LFS. Навіть якщо ви дійсно зіткнулися з проблемою, пов'язаною з розробниками, спільнота LFS може допомогти виділити інформацію, необхідну розробникам, і скласти відповідний звіт.

Якщо ви мусите задати питання безпосередньо через канал підтримки upstream, ви повинні припинити врахувати, що багато проектів upstream мають канали підтримки, відокремлені від системи відстеження помилок. Повідомлення про «помилки» для задавання питань вважаються недійсними і можуть дратувати розробників upstream цих проектів.

Якщо під час роботи з цією книгою виникла проблема або питання, перегляньте сторінку з часто задаваними питаннями за адресою <https://www.linuxfromscratch.org/faq/#generalfaq>. Часто відповіді на питання вже є там. Якщо відповіді на ваше питання немає на цій сторінці, спробуйте знайти джерело проблеми. Наступна підказка допоможе вам у вирішенні проблеми: <https://www.linuxfromscratch.org/hints/downloads/files/errors.txt>.

Якщо ви не можете знайти свою проблему в списку часто задаваних питань, пошукайте її в списках розсилки на <https://www.linuxfromscratch.org/search.html>.

У нас також є чудова спільнота LFS, яка готова надати допомогу через списки розсилки та IRC (див. розділ 1.4 «Ресурси» цієї книги). Однак щодня ми отримуємо кілька запитань щодо підтримки, і на багато з них можна було б легко відповісти, переглянувши FAQ або спочатку здійснивши пошук у списках розсилки. Тому, щоб ми могли надати найкращу допомогу, спочатку слід самостійно здійснити деякі пошуки. Це дозволить нам зосередитися на більш незвичайних потребах у підтримці. Якщо ваші пошуки не дають результату, будь ласка, включіть всю відповідну інформацію (зазначену нижче) у ваш запит про допомогу.

1.5.1. Що варто згадати

Окрім короткого пояснення проблеми, будь-який запит про допомогу повинен містити такі необхідні елементи:

- Версія книги, що використовується (в даному випадку 12.3)
- Дистрибутив хоста та версія, що використовуються для створення LFS
- Вихідні дані скрипта Host System Requirements
- Пакет або розділ, в якому виникла проблема
- Точне повідомлення про помилку або чіткий опис проблеми
- Зазначте, чи відхилялися ви від інструкцій книги



Note

Відхилення від цієї книги не означає, що ми не допоможемо вам. Адже LFS — це питання особистих уподобань. Відкритість щодо будь-яких змін у встановленій процедурі допомагає нам оцінити та визначити можливі причини вашої проблеми.

1.5.2. Проблеми зі скриптом `Configure`

Якщо під час виконання скрипту `configure` щось піде не так, перегляньте файл `config.log`. У ньому можуть міститися помилки, що виникли під час конфігурації, які не були виведені на екран. Якщо вам знадобиться допомога, обов'язково додайте відповідні рядки з цього файлу.

1.5.3. Проблеми компіляції

Щоб визначити причину проблем компіляції, корисно переглянути як вивід на екран, так і вміст різних файлів. Вивід на екран від скрипту `configure` та від команди `make` може бути корисним. Не обов'язково включати весь вивід, але варто додати всю відповідну інформацію. Нижче наведено приклад інформації, яку слід включити з екранного виводу `make`.

```
gcc -D ALIASPATH=\"/mnt/lfs/usr/share/locale:.\"
-D LOCALEDIR=\"/mnt/lfs/usr/share/locale\"
-D LIBDIR=/mnt/lfs/usr/lib\
-D INCLUDEDIR=\"/mnt/lfs/usr/include\" -D HAVE_CONFIG_H -I. -I.
-g -O2 -c getopt1.c
gcc -g -O2 -static -o make ar.o arscan.o commands.o dir.o
expand.o file.o function.o getopt.o implicit.o job.o main.o
misc.o read.o remake.o rule.o signature.o variable.o vpath.o
default.o remote-stub.o version.o getopt.o
-lutil job.o: In function `load_too_high':
/lfs/tmp/make-3.79.1/job.c:1565: undefined reference
to `getloadavg'
collect2: ld returned 1 exit status
make[2]: *** [make] Error 1
```

```
make[2]: Leaving directory `/lfs/tmp/make-3.79.1'  
make[1]: *** [all-recursive] Error 1  
make[1]: Leaving directory `/lfs/tmp/make-3.79.1'  
make: *** [all-recursive-am] Error 2
```

У цьому випадку багато хто просто включив би нижню частину:

```
make [2]: *** [make] Error 1
```

Цієї інформації недостатньо для діагностики проблеми, оскільки вона лише вказує на те, що щось пішло не так, але не вказує, що саме пішло не так. Весь фрагмент, як у наведеному вище прикладі, слід зберегти, оскільки він містить команду, яка була виконана, та всі пов'язані з нею повідомлення про помилки.

Відмінна стаття про те, як просити про допомогу в Інтернеті, доступна за адресою <http://catb.org/~esr/faqs/smart-questions.html>. Прочитайте цей документ і дотримуйтесь порад. Це збільшить ймовірність отримання необхідної допомоги.

Частина II. Підготовка до побудови

Глава 2. Підготовка хост-системи

2.1. Вступ

У цьому розділі перевіряються інструменти хоста, необхідні для побудови LFS, і, якщо потрібно, встановлюються. Потім готується розділ, на якому буде розміщена система LFS. Ми створимо сам розділ, створимо на ньому файлову систему і змонтуюмо її.

2.2. Вимоги до хост-системи

2.2.1. Апаратне забезпечення

Редактори LFS рекомендують, щоб процесор мав щонайменше чотири ядра, а система — щонайменше 8 ГБ пам'яті. Старіші системи, які не відповідають цим вимогам, також працюватимуть, але час на створення пакетів буде значно довшим, ніж зазначено в документації.

2.2.2. Програмне забезпечення

Ваша хост-система повинна мати наступне програмне забезпечення з вказаними мінімальними версіями. Це не повинно бути проблемою для більшості сучасних дистрибутивів Linux. Також зверніть увагу, що багато дистрибутивів розміщують заголовки програмного забезпечення в окремих пакетах, часто у формі <назва-пакета>-devel або <назва-пакета>-dev. Обов'язково встановіть їх, якщо ваш дистрибутив їх надає.

Більш ранні версії перелічених програмних пакетів можуть працювати, але не були протестовані.

- **Bash-3.2** (/bin/sh повинен бути символічним або жорстким посиланням на bash)
- **Binutils-2.13.1** (версії вище 2.44 не рекомендуються, оскільки вони не були протестовані)
- **Bison-2.7** (/usr/bin/yacc повинен бути посиланням на bison або невеликим скриптом, що виконує bison)
- **Coreutils-8.1**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk повинен бути посиланням на gawk)
- **GCC-5.2**, включаючи компілятор C++, **g++** (версії вище 14.2.0 не рекомендуються, оскільки вони не були протестовані). Стандартні бібліотеки C і C++ (з заголовками) також повинні бути присутніми, щоб компілятор C++ міг створювати хостингові програми
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Ядро Linux-5.4**

Причиною вимог до версії ядра є те, що ми вказуємо цю версію під час компіляції glibc у розділах 5 та 8, тому обхідні шляхи для старіших ядер не вмикаються, а скомпільована glibc є дещо швидшою та меншою. Станом на грудень 2024 року 5.4 є найстарішою версією ядра, яка все ще підтримується розробниками ядра. Деякі версії ядра старіші за 5.4 можуть все ще підтримуватися сторонніми командами, але вони не вважаються офіційними версіями ядра; детальніше читайте на <https://kernel.org/category/releases.html>.

Якщо ядро хоста є старішим за 5.4, вам потрібно буде замінити ядро на більш сучасну версію. Це можна зробити двома способами. Спочатку перевірте, чи ваш постачальник Linux надає пакет ядра версії 5.4 або пізнішої. Якщо так, ви можете його встановити. Якщо ваш постачальник не пропонує прийнятного пакета ядра або ви не бажаєте його встановлювати, ви можете скомпілювати ядро самостійно. Інструкції щодо компіляції ядра та налаштування завантажувача (за умови, що хост використовує GRUB) наведено в розділі 10.

Ми вимагаємо, щоб ядро хоста підтримувало псевдотермінал UNIX 98 (PTY). Він повинен бути увімкнений на всіх настільних або серверних дистрибутивах з ядром Linux 5.4 або новішим. Якщо ви створюєте власне ядро хоста, переконайтесь, що CONFIG_UNIX98_PTYS встановлено на у в конфігурації ядра.

- **M4-1.4.10**
- **Make-4.0**
- **Patch-2.5.4**
- **Perl-5.8.8**
- **Python-3.4**
- **Sed-4.1.5**
- **Tar-1.22**
- **Texinfo-5.0**
- **Xz-5.0.0**



Important

Зверніть увагу, що згадані вище символічні посилання необхідні для побудови системи LFS за інструкціями, що містяться в цій книзі. Символьні посилання, що вказують на інше програмне забезпечення (таке як dash, mawk тощо), можуть працювати, але не тестиувалися і не підтримуються командою розробників LFS, і можуть вимагати відхилення від інструкцій або додаткових виправлень до деяких пакетів.

Щоб перевірити, чи має ваша хост-система всі необхідні версії та можливість компілювати програми, виконайте наступні команди:

```
cat > version-check.sh << "EOF"
#!/bin/bash
# A script to list version numbers of critical development tools

# If you have tools installed in other directories, adjust PATH here AND
# in ~lfs/.bashrc (section 4.4) as well.

LC_ALL=C
PATH=/usr/bin:/bin

bail() { echo "FATAL: $1"; exit 1; }
grep --version > /dev/null 2> /dev/null || bail "grep does not work"
sed '' /dev/null || bail "sed does not work"
sort /dev/null || bail "sort does not work"
```

```

ver_check()
{
    if ! type -p $2 &>/dev/null
    then
        echo "ERROR: Cannot find $2 ($1)"; return 1;
    fi
    v=$(($2 --version 2>&1 | grep -E -o '[0-9]+\.[0-9\.\.]+[a-z]*' | head -n1)
    if printf '%s\n' $3 $v | sort --version-sort --check &>/dev/null
    then
        printf "OK:      %-9s %-6s >= $3\n" "$1" "$v"; return 0;
    else
        printf "ERROR:      %-9s is TOO OLD ($3 or later required)\n" "$1";
        return 1;
    fi
}

ver_kernel()
{
    kver=$(uname -r | grep -E -o '^([0-9\.\.]+)')
    if printf '%s\n' $1 $kver | sort --version-sort --check &>/dev/null
    then
        printf "OK:  Linux Kernel $kver >= $1\n"; return 0;
    else
        printf "ERROR:  Linux Kernel ($kver) is TOO OLD ($1 or later required)\n"
    "$kver";
        return 1;
    fi
}

# Coreutils first because --version-sort needs Coreutils >= 7.0
ver_check Coreutils      sort      8.1 || bail "Coreutils too old, stop"
ver_check Bash            bash      3.2
ver_check Binutils       ld        2.13.1
ver_check Bison           bison     2.7
ver_check Diffutils      diff      2.8.1
ver_check Findutils      find      4.2.31
ver_check Gawk            gawk      4.0.1
ver_check GCC             gcc       5.2
ver_check "GCC (C++)"    g++       5.2
ver_check Grep            grep      2.5.1a
ver_check Gzip            gzip      1.3.12
ver_check M4              m4        1.4.10
ver_check Make            make      4.0
ver_check Patch           patch     2.5.4
ver_check Perl            perl      5.8.8
ver_check Python          python3   3.4
ver_check Sed              sed       4.1.5
ver_check Tar              tar       1.22
ver_check Texinfo         texi2any 5.0
ver_check Xz              xz       5.0.0
ver_kernel 5.4

```

```

if mount | grep -q 'devpts on /dev/pts' && [ -e /dev/ptmx ]
then echo "OK: Linux Kernel supports UNIX 98 PTY";
else echo "ERROR: Linux Kernel does NOT support UNIX 98 PTY"; fi

alias_check() {
    if $1 --version 2>&1 | grep -qi $2
    then printf "OK: %-4s is $2\n" "$1";
    else printf "ERROR: %-4s is NOT $2\n" "$1"; fi
}
echo "Aliases:"
alias_check awk GNU
alias_check yacc Bison
alias_check sh Bash

echo "Compiler check:"
if printf "int main(){}" | g++ -x c++ -
then echo "OK: g++ works";
else echo "ERROR: g++ does NOT work"; fi
rm -f a.out

if [ "$(nproc)" = "" ]; then
    echo "ERROR: nproc is not available or it produces empty output"
else
    echo "OK: nproc reports $(nproc) logical cores are available"
fi
EOF

bash version-check.sh

```

2.3. Етапи побудови LFS

LFS розроблений для побудови за один сеанс. Тобто, інструкції передбачають, що система не буде вимкнена під час процесу. Це не означає, що система має бути побудована за один раз. Проблема полягає в тому, що певні процедури повинні бути повторені після перезавантаження при відновленні LFS в різних точках.

2.3.1. Глави 1–4

В цих главах виконуються команди на хост-системі. При перезапуску переконайтесь в одному:

- Процедури, що виконуються як користувач `root` після розділу 2.4, повинні мати змінну середовища LFS, встановлену **ДЛЯ КОРИСТУВАЧА ROOT**.

2.3.2. Глави 5–6

- Розділ `/mnt/lfs` повинен бути змонтований.
- Ці два розділи необхідно виконувати як користувач `lfs`. Перед виконанням будь-якого завдання в цих розділах необхідно виконати команду `su - lfs`. Якщо ви цього не зробите, ви ризикуєте встановити пакети на хост і, можливо, зробити його непридатним для використання.

- Процедури, описані в Загальних інструкціях з компіляції, є надзвичайно важливими. Якщо ви сумніваєтесь, чи пакет було встановлено правильно, переконайтесь, що раніше розпакований архів tar було видалено, потім знову розпакуйте пакет і виконайте всі інструкції, наведені в цьому розділі.

2.3.3. Глави 7–10

- Розділ /mnt/lfs повинен бути змонтований.
- Деякі операції, від «Підготовка віртуальних файлових систем ядра» до «Вхід в середовище chroot», повинні бути виконані як користувач `root`, з встановленою змінною середовища LFS для користувача `root`.
- При вході в chroot, змінна середовища LFS повинна бути встановлена для `root`. Змінна LFS не використовується після входу в середовище chroot.
- Віртуальні файлові системи повинні бути змонтовані. Це можна зробити до або після входу в chroot, перейшовши на віртуальний термінал хоста і, як `root`, виконавши команди, описані в розділі 7.3.1, «Монтування та заповнення /dev» та розділі 7.3.2, «Монтування віртуальних файлових систем ядра».

2.4. Створення нового розділу

Як і більшість інших операційних систем, LFS зазвичай встановлюється на виділений розділ. Рекомендований підхід до створення системи LFS полягає у використанні доступного порожнього розділу або, якщо у вас є достатньо нерозподіленого простору, у створенні такого розділу.

Мінімальна система вимагає розділу розміром близько 10 гігабайт (ГБ). Цього достатньо для зберігання всіх архівів з вихідним кодом та компіляції пакетів. Однак, якщо система LFS призначена для використання в якості основної системи Linux, ймовірно, буде встановлено додаткове програмне забезпечення, яке вимагатиме додаткового простору. Розділ розміром 30 ГБ є розумним розміром, що забезпечує можливість розширення. Сама система LFS не займе стільки місця. Значна частина цього обсягу потрібна для забезпечення достатнього вільного тимчасового сховища, а також для додавання додаткових можливостей після завершення LFS. Крім того, компіляція пакетів може вимагати багато дискового простору, який буде звільнено після встановлення пакета.

Оскільки для процесів компіляції не завжди вистачає оперативної пам'яті (RAM), доцільно використовувати невеликий розділ диска як область підкачки. Вона використовується ядром для зберігання рідко використовуваних даних і звільнення більшої кількості пам'яті для активних процесів. Розділ підкачки для системи LFS може бути таким самим, як і той, що використовується головним комп'ютером, і в цьому випадку не потрібно створювати інший.

Запустіть програму для розділення диска, таку як `cfdisk` або `fdisk`, з опцією командного рядка, що вказує жорсткий диск, на якому буде створено новий розділ, наприклад `/dev/sda` для основного диска. Створіть розділ для Linux та розділ підкачки, якщо це необхідно. Якщо ви ще не знаєте, як користуватися цими програмами, зверніться до `cfdisk(8)` або `fdisk(8)`.



Note

Для досвідчених користувачів можливі інші схеми розділення. Нова система LFS може бути розміщена на програмному масиві *RAID* або логічному томі *LVM*. Однак деякі з цих опцій вимагають *initramfs*, що є просунутою темою. Ці методи розділення не рекомендуються для користувачів, які вперше використовують LFS.

Запам'ятайте позначення нового розділу (наприклад, *sda5*). У цій книзі він буде називатися розділом LFS. Також запам'ятайте позначення розділу *swap*. Ці імена знадобляться пізніше для файлу */etc/fstab*.

2.4.1. Інші проблеми з розділами

Запити щодо порад стосовно розділення системи часто публікуються на списках розсилки LFS. Це дуже суб'єктивна тема. За замовчуванням у більшості дистрибутивів використовується весь диск, за винятком одного невеликого розділу підкачки. Це не є оптимальним для LFS з кількох причин. Це зменшує гнучкість, ускладнює обмін даними між декількома дистрибутивами або збірками LFS, робить резервне копіювання більш трудомістким і може привести до марнування дискового простору через неефективний розподіл структур файлової системи.

2.4.1.1. Кореневий розділ

Кореневий розділ LFS (не плутати з каталогом */root*) розміром двадцять гігабайт є хорошим компромісом для більшості систем. Він забезпечує достатньо місця для побудови LFS і більшої частини BLFS, але є достатньо малим, щоб можна було легко створити кілька розділів для експериментів.

2.4.1.2. Розділ підкачки

Більшість дистрибутивів автоматично створюють розділ підкачки. Зазвичай рекомендується, щоб розмір розділу підкачки був приблизно вдвічі більшим за обсяг фізичної оперативної пам'яті, проте це рідко буває необхідним. Якщо простір на диску обмежений, обмежте розмір розділу підкачки двома гігабайтами і стежте за обсягом підкачки з диска.

Якщо ви хочете використовувати функцію гібернації (suspend-to-disk) Linux, вона записує вміст оперативної пам'яті на розділ підкачки перед вимкненням машини. У цьому випадку розмір розділу підкачки повинен бути принаймні таким же великим, як і встановлена оперативна пам'ять системи.

Підкачка ніколи не є хорошим явищем. У випадку механічних жорстких дисків зазвичай можна визначити, чи відбувається підкачка даних, просто прослухавши активність диска та спостерігаючи за реакцією системи на команди. У випадку SSD ви не зможете почути підкачку даних, але ви можете визначити, скільки місця для обміну даними використовується, запустивши програми **top** або **free**. Якщо можливо, слід уникати використання SSD для розділу підкачки. Першою реакцією на підкачку даних має бути перевірка на наявність необґрунтованих команд, таких як спроба редагувати файл розміром п'ять гігабайт. Якщо підкачка стає нормальним явищем, найкращим рішенням буде придбати більше оперативної пам'яті для вашої системи.

2.4.1.3. Розділ Grub Bios

Якщо завантажувальний диск було розділено за допомогою таблиці розділів GUID (GPT), то необхідно створити невеликий розділ, зазвичай розміром 1 МБ, якщо він ще не існує. Цей розділ не форматується, але повинен бути доступним для використання GRUB під час встановлення завантажувача. Зазвичай цей розділ має позначку «BIOS Boot», якщо використовується `fdisk`, або код `EF02`, якщо використовується команда `gdisk`.



Note

Розділ Grub Bios повинен знаходитися на диску, який BIOS використовує для завантаження системи. Це не обов'язково диск, на якому знаходиться кореневий розділ LFS. Диски в системі можуть використовувати різні типи таблиць розділів. Необхідність розділу Grub Bios залежить тільки від типу таблиці розділів завантажувального диска.

2.4.1.4. Додаткові розділи для зручності

Є кілька інших розділів, які не є обов'язковими, але їх слід враховувати при проєктуванні структури диска. Наведений нижче перелік не є вичерпним, але може слугувати орієнтиром.

- `/boot` – Наполегливо рекомендується. Використовуйте цей розділ для зберігання ядер та іншої інформації для завантаження. Щоб мінімізувати потенційні проблеми із завантаженням на великих дисках, зробіть цей розділ першим фізичним розділом на вашому першому дисководі. Розмір розділу 200 мегабайт є достатнім.
- `/boot/efi` – Розділ системи EFI, який необхідний для завантаження системи з UEFI. Детальніше читайте на сторінці BLFS.
- `/home` – Наполегливо рекомендується. Спільне використання `home` каталогу та налаштувань користувача в декількох дистрибутивах або збірках LFS. Розмір зазвичай досить великий і залежить від доступного місця на диску.
- `/usr` – У LFS, `/bin`, `/lib` та `/sbin` є символічними посиланнями на їхні відповідники в `/usr`. Отже, `/usr` містить усі бінарні файли, необхідні для роботи системи. Для LFS окремий розділ для `/usr` зазвичай не потрібен. Якщо ви все ж створюєте його, ви повинні зробити розділ достатньо великим, щоб вмістити всі програми та бібліотеки в системі. Розділ `root` може бути дуже малим (можливо, всього один гігабайт) у цій конфігурації, тому він підходить для тонкого клієнта або бездискової робочої станції (де `/usr` монтується з віддаленого сервера). Однак слід пам'ятати, що для завантаження системи з окремим розділом `/usr` буде потрібен `initramfs` (не висвітлений в LFS).
- `/opt` – Цей каталог є найбільш корисним для BLFS, де можна встановити кілька великих пакетів, таких як KDE або Texlive, без будовування файлів в ієрархію `/usr`. Якщо він використовується, зазвичай достатньо від 5 до 10 гігабайт.
- `/tmp` – Окремий розділ `/tmp` зустрічається рідко, але він корисний при налаштуванні тонкого клієнта. Якщо цей розділ використовується, зазвичай його розмір не повинен перевищувати декількох гігабайтів. Якщо у вас достатньо оперативної пам'яті, ви можете змонтувати `tmpfs` на `/tmp`, щоб пришвидшити доступ до тимчасових файлів.

- /usr/src – Цей розділ дуже корисний для зберігання вихідних файлів BLFS та їх спільного використання в складанні LFS. Він також може використовуватися як місце для складання пакетів BLFS. Досить великий розділ розміром 30-50 гігабайт забезпечить достатньо місця.

Будь-який окремий розділ, який ви хочете автоматично монтувати під час запуску системи, повинен бути вказаний у файлі /etc/fstab. Детальніше про те, як вказувати розділи, буде розглянуто в розділі 10.2, «Створення файлу /etc/fstab».

2.5. Створення файлової системи на розділі

Розділ — це просто діапазон секторів на диску, обмежений межами, встановленими в таблиці розділів. Перш ніж операційна система зможе використовувати розділ для зберігання будь-яких файлів, розділ необхідно відформатувати, щоб він містив файлову систему, яка зазвичай складається з мітки, блоків каталогів, блоків даних та схеми індексації для пошуку певного файлу за запитом. Файлова система також допомагає ОС відстежувати вільний простір на розділі, резервувати необхідні сектори при створенні нового файлу або розширенні існуючого файлу, а також переробляти вільні сегменти даних, створені при видаленні файлів. Вона також може забезпечувати підтримку надмірності даних та відновлення після помилок.

LFS може використовувати будь-яку файлову систему, яку розпізнає ядро Linux, але найпоширенішими типами є ext3 та ext4. Вибір правильної файлової системи може бути складним; він залежить від характеристик файлів та розміру розділу. Наприклад:

ext2

підходить для невеликих розділів, які оновлюються нечасто, таких як /boot.

ext3

є оновленою версією ext2, яка включає журнал, що допомагає відновити стан розділу в разі некоректного вимкнення системи. Він зазвичай використовується як файлова система загального призначення.

ext4

є останньою версією сімейства файлових систем ext. Він надає кілька нових можливостей, включаючи наносекундні мітки часу, створення та використання дуже великих файлів (до 16 ТБ) та підвищення швидкості.

Інші файлові системи, включаючи FAT32, NTFS, JFS та XFS, корисні для спеціальних цілей. Більше інформації про ці файлові системи та багато інших можна знайти на сайті https://en.wikipedia.org/wiki/Comparison_of_file_systems.

LFS припускає, що коренева файлова система (/) має тип ext4. Щоб створити файлову систему ext4 на розділі LFS, виконайте наступну команду:

```
mkfs -v -t ext4 /dev/<xxx>
```

Замініть <xxx> на назву розділу LFS.

Якщо ви використовуєте існуючий розділ swap, форматувати його не потрібно. Якщо було створено новий розділ swap, його необхідно ініціалізувати за допомогою цієї команди:

```
mkswap /dev/<yyy>
```

Замініть <yyy> на назву розділу підкачки.

2.6. Встановлення змінної \$LFS та значення umask

У цій книзі кілька разів буде використовуватися змінна середовища LFS. Ви повинні переконатися, що ця змінна завжди визначена протягом усього процесу побудови LFS. Вона повинна бути встановлена на ім'я каталогу, в якому ви будете будувати свою систему LFS - ми будемо використовувати `/mnt/lfs` як приклад, але ви можете вибрати будь-яке ім'я каталогу, яке вам подобається. Якщо ви будуєте LFS на окремому розділі, цей каталог буде точкою монтування для розділу. Виберіть розташування каталогу і встановіть змінну за допомогою наступної команди:

```
export LFS=/mnt/lfs
```

Наявність цієї змінної є корисною, оскільки команди, такі як `mkdir -v $LFS/tools`, можна вводити буквально. Оболонка автоматично замінить «\$LFS» на «/mnt/lfs» (або будь-яке інше значення, яке було встановлено для змінної) під час обробки командного рядка.

Тепер встановіть маску створення файлів (umask) на 022 на випадок, якщо дистрибутив хоста використовує інше значення за замовчуванням:

```
umask 022
```

Встановлення umask на 022 гарантує, що новстворені файли та каталоги будуть доступні для запису тільки їх власнику, але будуть доступні для читання та пошуку (тільки для каталогів) будь-яким користувачем (якщо припустити, що системний виклик `open(2)` використовує режими за замовчуванням, нові файли отримають режим доступу 644, а каталоги — режим 755). Надто дозвільний режим за замовчуванням може створити діри в безпеці системи LFS, а надто обмежувальний режим за замовчуванням може спричинити дивні проблеми під час побудови або використання системи LFS.



Caution

Не забувайте перевіряти, чи встановлено `LFS` і чи встановлено `umask` на 022, коли ви залишаєте і знову входите в поточне робоче середовище (наприклад, коли виконуєте `su` для `root` або іншого користувача). Перевірте, чи правильно встановлено змінну `LFS` за допомогою:

```
echo $LFS
```

Переконайтесь, що у вихідних даних вказано шлях до місця побудови вашої системи LFS, який є `/mnt/lfs`, якщо ви дотримувалися наведеного прикладу.

Перевірте, чи правильно налаштовано `umask`, за допомогою команди:

```
umask
```

Результатом може бути 0022 або 022 (кількість провідних нулів залежить від дистрибутива хоста).

Якщо будь-який результат цих двох команд є неправильним, використовуйте команди, наведені раніше на цій сторінці, щоб встановити `$LFS` на правильну назву каталогу та встановити `umask` на 022.



Note

Один із способів забезпечити правильне встановлення змінної `LFS` та `umask` — це редагувати файл `.bash_profile` як у вашому особистому домашньому каталозі, так і в `/root/.bash_profile` та ввести команди `export` та `umask`, наведені вище. Крім того, оболонкою, вказаною у файлі `/etc/passwd` для всіх користувачів, яким потрібна змінна `LFS`, має бути `bash`, щоб забезпечити включення файлу `.bash_profile` до процесу входу в систему.

Іншим важливим моментом є метод, який використовується для входу в систему хоста. Якщо вхід здійснюється через графічний дисплейний менеджер, файл `.bash_profile` користувача зазвичай не використовується при запуску віртуального терміналу. У цьому випадку додайте команди до файлу `.bashrc` для користувача та `root`. Крім того, деякі дистрибутиви використовують тест «`if`» і не виконують решту інструкцій `.bashrc` для неінтерактивного виклику `bash`. Обов'язково розмістіть команди перед тестом для неінтерактивного використання.

2.7. Монтування нового розділу

Тепер, коли файлова система створена, розділ потрібно змонтувати, щоб хост-система могла отримати до нього доступ. У цій книзі припускається, що файлова система змонтована в каталозі, вказаному змінною середовища `LFS`, описаною в попередньому розділі.

Строго кажучи, не можна «підключити розділ». Підключається файлова система, вбудована в цей розділ. Але оскільки один розділ не може містити більше однієї файлової системи, люди часто говорять про розділ і пов'язану з ним файлову систему, як про одне і те ж.

Створіть точку монтування та змонтуйте файлову систему `LFS` за допомогою таких команд:

```
mkdir -pv $LFS
mount -v -t ext4 /dev/<xxx> $LFS
```

Замініть `<xxx>` на ім'я розділу `LFS`.

Якщо ви використовуєте кілька розділів для `LFS` (наприклад, один для `/`, а інший для `/home`), змонтуйте їх таким чином:

```
mkdir -pv $LFS
mount -v -t ext4 /dev/<xxx> $LFS
mkdir -v $LFS/home
mount -v -t ext4 /dev/<yyy> $LFS/home
```

Замініть `<xxx>` та `<yyy>` відповідними назвами розділів.

Встановіть власника та режим доступу до каталогу `$LFS` (тобто кореневого каталогу в новоствореній файловій системі для системи `LFS`) на `root` та `755`, якщо дистрибутив хоста налаштований на використання іншого значення за замовчуванням для `mkfs`:

```
chown root:root $LFS
chmod 755 $LFS
```

Переконайтесь, що цей новий розділ не змонтований з надто обмежувальними правами доступу (такими як опції `nosuid` або `nodev`). Виконайте команду `mount` без будь-яких параметрів, щоб побачити, які опції встановлені для змонтованого розділу LFS. Якщо встановлено `nosuid` та/або `nodev`, розділ необхідно змонтовувати заново.



Warning

Вищезазначені інструкції передбачають, що ви не будете перезавантажувати комп'ютер протягом усього процесу LFS. Якщо ви вимкнете систему, вам доведеться або перемонтувати розділ LFS кожного разу, коли ви перезапускаєте процес побудови, або змінити файл `/etc/fstab` хост-системи, щоб він автоматично перемонтувався під час перезавантаження. Наприклад, ви можете додати подібний рядок до файлу `/etc/fstab`:

```
/dev/<xxx> /mnt/lfs ext4 defaults 1 1
```

Якщо ви використовуєте додаткові опціональні розділи, обов'язково додайте їх також.

Якщо ви використовуєте розділ `swap`, переконайтесь, що він увімкнений за допомогою команди `swapon`:

```
/sbin/swapon -v /dev/<zzz>
```

Замініть `<zzz>` на назву `swap` розділу.

Тепер, коли новий розділ LFS готовий до роботи, настав час завантажити пакети.

Глава 3. Пакети та патчі

3.1. Вступ

Цей розділ містить перелік пакетів, які необхідно завантажити для побудови базової системи Linux. Вказані номери версій відповідають версіям програмного забезпечення, які, як відомо, працюють, і ця книга базується на їх використанні. Ми наполегливо не рекомендуємо використовувати інші версії, оскільки команди побудови для однієї версії можуть не працювати з іншою версією, якщо інша версія не вказана в виправленні LFS або рекомендації з безпеки. Новітні версії пакетів також можуть мати проблеми, які вимагають обхідних рішень. Ці обхідні рішення будуть розроблені та стабілізовані в розробницькій версії книги.

Для деяких пакетів архів релізу та архів знімка репозиторію (Git або SVN) для цього релізу можуть бути опубліковані з подібними або навіть ідентичними іменами файлів. Але архів релізу може містити деякі файли, які є необхідними, незважаючи на те, що вони не зберігаються в репозиторії (наприклад, скрипт конфігурації, згенерований autoconf), на додаток до вмісту відповідного знімка репозиторію. У книзі використовуються архівні файли релізів, коли це можливо. Використання знімка репозиторію замість архівного файла релізу, зазначеного в книзі, може спричинити проблеми.

Місця завантаження можуть бути не завжди доступними. Якщо місце завантаження змінилося з моменту публікації цієї книги, Google (<https://www.google.com/>) надає корисний пошуковий механізм для більшості пакетів. Якщо цей пошук не дав результатів, спробуйте один з альтернативних способів завантаження на <https://www.linuxfromscratch.org/lfs/mirrors.html#files>.

Завантажені пакети та виправлення потрібно зберігати в місці, яке буде зручно доступним протягом усього процесу збирання. Також потрібен робочий каталог для розпакування джерел та їхньої збиранки. `$LFS/sources` можна використовувати як місце для зберігання архівів та виправлень, так і як робочий каталог. Використовуючи цей каталог, необхідні елементи будуть розміщені на розділі LFS і будуть доступні на всіх етапах процесу збиранки.

Щоб створити цей каталог, перед початком сеансу завантаження виконайте наступну команду як користувач `root`:

```
mkdir -v $LFS/sources
```

Зробіть цей каталог доступним для запису та sticky. «Sticky» означає, що навіть якщо кілька користувачів мають права на запис у каталозі, тільки власник файла може видалити файл із sticky каталогу. Наступна команда увімкне режими запису та sticky:

```
chmod -v a+wt $LFS/sources
```

Існує кілька способів отримати всі необхідні пакети та виправлення для побудови LFS:

- Файли можна завантажити окремо, як описано в наступних двох розділах.
- Для стабільних версій книги архів усіх необхідних файлів можна завантажити з одного з дзеркальних сайтів, перелічених на <https://www.linuxfromscratch.org/mirrors.html#files>.
- Файли можна завантажити за допомогою `wget` і `wget-list`, як описано нижче.

Щоб завантажити всі пакети та виправлення, використовуючи `wget-list-sysv` як вхідні дані для команди `wget`, скористайтеся наступною командою:

```
wget --input-file=wget-list-sysv --continue --directory-prefix=$LFS/sources
```

Крім того, починаючи з LFS-7.0, існує окремий файл `md5sums`, який можна використовувати для перевірки наявності всіх необхідних пакетів перед продовженням роботи. Помістіть цей файл у `$LFS/sources` і виконайте:

```
pushd $LFS/sources
    md5sum -c md5sums
popd
```

Цю перевірку можна використовувати після отримання необхідних файлів будь-яким із перерахованих вище методів.

Якщо пакети та виправлення завантажуються користувачем, що не є `root`, ці файли будуть належати цьому користувачеві. Файлова система записує власника за його UID, а UID звичайного користувача із дистрибутиву хоста не призначаються в LFS. Тому файли залишаться власністю невідомого UID в кінцевій системі LFS. Якщо ви не будете призначати той самий UID для свого користувача в системі LFS, змініть в `root` ласників цих файлів на зараз, щоб уникнути цієї проблеми:

```
chown root:root $LFS/sources/*
```

3.2. Всі пакети



Note

Перед завантаженням пакетів прочитайте *рекомендації з безпеки*, щоб з'ясувати, чи слід використовувати новішу версію будь-якого пакета щоб уникнути безпекових вразливостей.

Джерела верхнього рівня можуть видаляти старі випуски, особливо якщо ці випуски містять безпекові вразливості. Якщо один із наведених нижче URL-адрес недоступний, спочатку прочитайте рекомендації з безпеки, щоб з'ясувати, чи слід використовувати новішу версію (з виправленою вразливістю). Якщо ні, спробуйте завантажити видалений пакет з дзеркала. Хоча стару версію можна завантажити з дзеркала, навіть якщо вона була видалена через вразливість, не рекомендується використовувати версію, яка, як відомо, є вразливою, під час побудови вашої системи.

Завантажте або іншим чином отримайте наступні пакети:

- **Acl (2.3.2) - 363 KB:**

Home page: <https://savannah.nongnu.org/projects/acl>

Download: <https://download.savannah.gnu.org/releases/acl/acl-2.3.2.tar.xz>

MD5 sum: 590765dee95907dbc3c856f7255bd669

- **Attr (2.5.2) - 484 KB:**

Home page: <https://savannah.nongnu.org/projects/attr>

Download: <https://download.savannah.gnu.org/releases/attr/attr-2.5.2.tar.gz>

MD5 sum: 227043ec2f6ca03c0948df5517f9c927

- **Autoconf (2.72) - 1,360 KB:**

Home page: <https://www.gnu.org/software/autoconf/>

Download: <https://ftp.gnu.org/gnu/autoconf/autoconf-2.72.tar.xz>

MD5 sum: 1be79f7106ab6767f18391c5e22be701

- **Automake (1.17) - 1,614 KB:**

Home page: <https://www.gnu.org/software/automake/>

Download: <https://ftp.gnu.org/gnu/automake/automake-1.17.tar.xz>

MD5 sum: 7ab3a02318fee6f5bd42adfc369abf10

- **Bash (5.2.37) - 10,868 KB:**

Home page: <https://www.gnu.org/software/bash/>

Download: <https://ftp.gnu.org/gnu/bash/bash-5.2.37.tar.gz>

MD5 sum: 9c28f21ff65de72ca329c1779684a972

- **Bc (7.0.3) - 464 KB:**

Home page: <https://git.gavinhoward.com/gavin/bc>

Download: <https://github.com/gavinhoward/bc/releases/download/7.0.3/bc-7.0.3.tar.xz>

MD5 sum: ad4db5a0eb4fdbb3f6813be4b6b3da74

- **Binutils (2.44) - 26,647 KB:**

Home page: <https://www.gnu.org/software/binutils/>

Download: <https://sourceware.org/pub/binutils/releases/binutils-2.44.tar.xz>

MD5 sum: 49912ce774666a30806141f106124294

- **Bison (3.8.2) - 2,752 KB:**

Home page: <https://www.gnu.org/software/bison/>

Download: <https://ftp.gnu.org/gnu/bison/bison-3.8.2.tar.xz>

MD5 sum: c28f119f405a2304ff0a7ccdcc629713

- **Bzip2 (1.0.8) - 792 KB:**

Download: <https://www.sourceware.org/pub/bzip2/bzip2-1.0.8.tar.gz>

MD5 sum: 67e051268d0c475ea773822f7500d0e5

- **Check (0.15.2) - 760 KB:**

Home page: <https://libcheck.github.io/check>

Download: <https://github.com/libcheck/check/releases/download/0.15.2/check-0.15.2.tar.gz>

MD5 sum: 50fcacfecde5a380415b12e9c574e0b2

- **Coreutils (9.6) - 5,991 KB:**

Home page: <https://www.gnu.org/software/coreutils/>

Download: <https://ftp.gnu.org/gnu/coreutils/coreutils-9.6.tar.xz>

MD5 sum: 0ed6cc983fe02973bc98803155cc1733

- **DejaGNU (1.6.3) - 608 KB:**

Home page: <https://www.gnu.org/software/dejagnu/>

Download: <https://ftp.gnu.org/gnu/dejagnu/dejagnu-1.6.3.tar.gz>

MD5 sum: 68c5208c58236eba447d7d6d1326b821

- **Diffutils (3.11) - 1,881 KB:**

Home page: <https://www.gnu.org/software/diffutils/>

Download: <https://ftp.gnu.org/gnu/diffutils/diffutils-3.11.tar.xz>

MD5 sum: 75ab2bb7b5ac0e3e10cece85bd1780c2

- **E2fsprogs (1.47.2) - 9,763 KB:**

Home page: <https://e2fsprogs.sourceforge.net/>

Download: <https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.47.2/e2fsprogs-1.47.2.tar.gz>

MD5 sum: 752e5a3ce19aea060d8a203f2fae9baa

- **Elfutils (0.192) - 11,635 KB:**

Home page: <https://sourceware.org/elfutils/>

Download: <https://sourceware.org/ftp/elfutils/0.192/elfutils-0.192.tar.bz2>

MD5 sum: a6bb1efc147302fcf15b5c2b827f186a

- **Expat (2.6.4) - 476 KB:**

Home page: <https://libexpat.github.io/>

Download: <https://prdownloads.sourceforge.net/expat/expat-2.6.4.tar.xz>

MD5 sum: 101fe3e320a2800f36af8cf4045b45c7

- **Expect (5.45.4) - 618 KB:**

Home page: <https://core.tcl.tk/expect/>

Download: <https://prdownloads.sourceforge.net/expect/expect5.45.4.tar.gz>

MD5 sum: 00fce8de158422f5ccd2666512329bd2

- **File (5.46) - 1,283 KB:**

Home page: <https://www.darwinsky.com/file/>

Download: <https://astron.com/pub/file/file-5.46.tar.gz>

MD5 sum: 459da2d4b534801e2e2861611d823864

- **Findutils (4.10.0) - 2,189 KB:**

Home page: <https://www.gnu.org/software/findutils/>

Download: <https://ftp.gnu.org/gnu/findutils/findutils-4.10.0.tar.xz>

MD5 sum: 870cf71c07d37ebe56f9f4aaf4ad872

- **Flex (2.6.4) - 1,386 KB:**

Home page: <https://github.com/westes/flex>

Download: <https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz>

MD5 sum: 2882e3179748cc9f9c23ec593d6adc8d

- **Flit-core (3.11.0) - 51 KB:**

Home page: <https://pypi.org/project/flit-core/>

Download: https://pypi.org/packages/source/f/flit-core/flit_core-3.11.0.tar.gz

MD5 sum: 6d677b1acef1769c4c7156c7508e0dbd

- **Gawk (5.3.1) - 3,428 KB:**

Home page: <https://www.gnu.org/software/gawk/>

Download: <https://ftp.gnu.org/gnu/gawk/gawk-5.3.1.tar.xz>

MD5 sum: 4e9292a06b43694500e0620851762eec

- **GCC (14.2.0) - 90,144 KB:**

Home page: <https://gcc.gnu.org/>

Download: <https://ftp.gnu.org/gnu/gcc/gcc-14.2.0/gcc-14.2.0.tar.xz>

MD5 sum: 2268420ba02dc01821960e274711bde0

- **GDBM (1.24) - 1,168 KB:**

Home page: <https://www.gnu.org/software/gdbm/>

Download: <https://ftp.gnu.org/gnu/gdbm/gdbm-1.24.tar.gz>

MD5 sum: c780815649e52317be48331c1773e987

- **Gettext (0.24) - 8,120 KB:**

Home page: <https://www.gnu.org/software/gettext/>

Download: <https://ftp.gnu.org/gnu/gettext/gettext-0.24.tar.xz>

MD5 sum: 87aea3013802a3c60fa3feb5c7164069

- **Glibc (2.41) - 18,892 KB:**

Home page: <https://www.gnu.org/software/libc/>

Download: <https://ftp.gnu.org/gnu/glibc/glibc-2.41.tar.xz>

MD5 sum: 19862601af60f73ac69e067d3e9267d4



Note

Розробники Glibc підтримують гілку Git, що містить виправлення, які вважаються гідними для Glibc-2.41, але, на жаль, були розроблені після випуску Glibc-2.41. Редактори LFS видауть рекомендацію з безпеки, якщо будь-яке виправлення безпеки буде додано до гілки, але жодних дій щодо інших новододаних виправлень вживатися не буде. Ви можете самостійно переглянути виправлення та включити деякі з них, якщо вважаєте їх важливими.

- **GMP (6.3.0) - 2,046 KB:**

Home page: <https://www.gnu.org/software/gmp/>

Download: <https://ftp.gnu.org/gnu/gmp/gmp-6.3.0.tar.xz>

MD5 sum: 956dc04e864001a9c22429f761f2c283

- **Gperf (3.1) - 1,188 KB:**

Home page: <https://www.gnu.org/software/gperf/>

Download: <https://ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz>

MD5 sum: 9e251c0a618ad0824b51117d5d9db87e

- **Grep (3.11) - 1,664 KB:**

Home page: <https://www.gnu.org/software/grep/>

Download: <https://ftp.gnu.org/gnu/grep/grep-3.11.tar.xz>

MD5 sum: 7c9bbd74492131245f7cdb291fa142c0

- **Groff (1.23.0) - 7,259 KB:**

Home page: <https://www.gnu.org/software/groff/>

Download: <https://ftp.gnu.org/gnu/groff/groff-1.23.0.tar.gz>

MD5 sum: 5e4f40315a22bb8a158748e7d5094c7d

- **GRUB (2.12) - 6,524 KB:**

Home page: <https://www.gnu.org/software/grub/>
Download: <https://ftp.gnu.org/gnu/grub/grub-2.12.tar.xz>
MD5 sum: 60c564b1bdc39d8e43b3aab4bc0fb140

- **Gzip (1.13) - 819 KB:**

Home page: <https://www.gnu.org/software/gzip/>
Download: <https://ftp.gnu.org/gnu/gzip/gzip-1.13.tar.xz>
MD5 sum: d5c9fc9441288817a4a0be2da0249e29

- **Iana-Etc (20250123) - 591 KB:**

Home page: <https://www.iana.org/protocols>
Download: <https://github.com/Mic92/iana-etc/releases/download/20250123/iana-etc-20250123.tar.gz>
MD5 sum: f8a0ebdc19a5004cf42d8bdcf614fa5d

- **Inetutils (2.6) - 1,724 KB:**

Home page: <https://www.gnu.org/software/inetutils/>
Download: <https://ftp.gnu.org/gnu/inetutils/inetutils-2.6.tar.xz>
MD5 sum: 401d7d07682a193960bcdcafd03de94

- **Intltool (0.51.0) - 159 KB:**

Home page: <https://freedesktop.org/wiki/Software/intltool>
Download: <https://launchpad.net/intltool/trunk/0.51.0/+download/intltool-0.51.0.tar.gz>
MD5 sum: 12e517cac2b57a0121cda351570f1e63

- **IPRoute2 (6.13.0) - 906 KB:**

Home page: <https://www.kernel.org/pub/linux/utils/net/iproute2/>
Download: <https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-6.13.0.tar.xz>
MD5 sum: 1603d25120d03feeaba9b360d03ffaec

- **Jinja2 (3.1.5) - 239 KB:**

Home page: <https://jinja.palletsprojects.com/en/3.1.x/>
Download: <https://pypi.org/packages/source/J/Jinja2/jinja2-3.1.5.tar.gz>
MD5 sum: 083d64f070f6f1b5f75971ae60240785

- **Kbd (2.7.1) - 1,438 KB:**

Home page: <https://kbd-project.org/>
Download: <https://www.kernel.org/pub/linux/utils/kbd/kbd-2.7.1.tar.xz>
MD5 sum: f15673d9f748e58f82fa50cff0d0fd20

- **Kmod (34) - 331 KB:**

Home page: <https://github.com/kmod-project/kmod>
Download: <https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-34.tar.xz>
MD5 sum: 3e6c5c9ad9c7367ab9c3cc4f08dfde62

- **Less (668) - 635 KB:**

Home page: <https://www.greenwoodsoftware.com/less/>
Download: <https://www.greenwoodsoftware.com/less/less-668.tar.gz>
MD5 sum: d72760386c5f80702890340d2f66c302

- **LFS-Bootscripts (20240825) - 33 KB:**

Download: <https://www.linuxfromscratch.org/lfs/downloads/12.3/lfs-bootscripts-20240825.tar.xz>

MD5 sum: 7b078c594a77e0f9cd53a0027471c3bc

- **Libcap (2.73) - 191 KB:**

Home page: <https://sites.google.com/site/fullycapable/>

Download: <https://www.kernel.org/pub/linux/libs/security/linux-prives/libcap2/libcap-2.73.tar.xz>

MD5 sum: 0e186df9de9b1e925593a96684fe2e32

- **Libffi (3.4.7) - 1,362 KB:**

Home page: <https://sourceware.org/libffi/>

Download: <https://github.com/libffi/libffi/releases/download/v3.4.7/libffi-3.4.7.tar.gz>

MD5 sum: 696a1d483a1174ce8a477575546a5284

- **Libpipeline (1.5.8) - 1046 KB:**

Home page: <https://libpipeline.nongnu.org/>

Download: <https://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.8.tar.gz>

MD5 sum: 17ac6969b2015386bcb5d278a08a40b5

- **Libtool (2.5.4) - 1,033 KB:**

Home page: <https://www.gnu.org/software/libtool/>

Download: <https://ftp.gnu.org/gnu/libtool/libtool-2.5.4.tar.xz>

MD5 sum: 22e0a29df8af5fdde276ea3a7d351d30

- **Libxcrypt (4.4.38) - 612 KB:**

Home page: <https://github.com/besser82/libxcrypt/>

Download: <https://github.com/besser82/libxcrypt/releases/download/v4.4.38/libxcrypt-4.4.38.tar.xz>

MD5 sum: 1796a5d20098e9dd9e3f576803c83000

- **Linux (6.13.4) - 145,015 KB:**

Home page: <https://www.kernel.org/>

Download: <https://www.kernel.org/pub/linux/kernel/v6.x/linux-6.13.4.tar.xz>

MD5 sum: 13b9e6c29105a34db4647190a43d1810



Note

Ядро Linux оновлюється досить часто, багато разів через виявлення вразливостей безпеки. Можна використовувати останню доступну стабільну версію ядра, якщо на сторінці errata не вказано інше. Для користувачів з обмеженою швидкістю або дорогим трафіком, які бажають оновити ядро Linux, базову версію пакета та виправлення можна завантажити окремо. Це може заощадити час або кошти при подальшому оновленні до новішої версії в рамках одного мінорного випуску.

- **Lz4 (1.10.0) - 379 KB:**

Home page: <https://lz4.org/>

Download: <https://github.com/lz4/lz4/releases/download/v1.10.0/lz4-1.10.0.tar.gz>

MD5 sum: dead9f5f1966d9ae56e1e32761e4e675

- **M4 (1.4.19) - 1,617 KB:**

Home page: <https://www.gnu.org/software/m4/>

Download: <https://ftp.gnu.org/gnu/m4/m4-1.4.19.tar.xz>

MD5 sum: 0d90823e1426f1da2fd872df0311298d

- **Make (4.4.1) - 2,300 KB:**

Home page: <https://www.gnu.org/software/make/>

Download: <https://ftp.gnu.org/gnu/make/make-4.4.1.tar.gz>

MD5 sum: c8469a3713cbbe04d955d4ae4be23eeb

- **Man-DB (2.13.0) - 2,023 KB:**

Home page: <https://www.nongnu.org/man-db/>

Download: <https://download.savannah.gnu.org/releases/man-db/man-db-2.13.0.tar.xz>

MD5 sum: 97ab5f9f32914eef2062d867381d8cee

- **Man-pages (6.12) - 1,838 KB:**

Home page: <https://www.kernel.org/doc/man-pages/>

Download: <https://www.kernel.org/pub/linux/docs/man-pages/man-pages-6.12.tar.xz>

MD5 sum: 44de430a598605eaba3e36dd43f24298

- **MarkupSafe (3.0.2) - 21 KB:**

Home page: <https://palletsprojects.com/p/markupsafe/>

Download: <https://pypi.org/packages/source/M/MarkupSafe/markupsafe-3.0.2.tar.gz>

MD5 sum: cb0071711b573b155cc8f86e1de72167

- **Meson (1.7.0) - 2,241 KB:**

Home page: <https://mesonbuild.com>

Download: <https://github.com/mesonbuild/meson/releases/download/1.7.0/meson-1.7.0.tar.gz>

MD5 sum: c20f3e5ebbb007352d22f4fd6ceb925c

- **MPC (1.3.1) - 756 KB:**

Home page: <https://www.multiprecision.org/>

Download: <https://ftp.gnu.org/gnu/mpc/mpc-1.3.1.tar.gz>

MD5 sum: 5c9bc658c9fd0f940e8e3e0f09530c62

- **MPFR (4.2.1) - 1,459 KB:**

Home page: <https://www.mpfr.org/>

Download: <https://ftp.gnu.org/gnu/mpfr/mpfr-4.2.1.tar.xz>

MD5 sum: 523c50c6318dde6f9dc523bc0244690a

- **Ncurses (6.5) - 2,156 KB:**

Home page: <https://www.gnu.org/software/ncurses/>

Download: <https://invisible-mirror.net/archives/ncurses/ncurses-6.5.tar.gz>

MD5 sum: ac2d2629296f04c8537ca706b6977687

- **Ninja (1.12.1) - 235 KB:**

Home page: <https://ninja-build.org/>

Download: <https://github.com/ninja-build/ninja/archive/v1.12.1/ninja-1.12.1.tar.gz>

MD5 sum: 6288992b05e593a391599692e2f7e490

- **OpenSSL (3.4.1) - 17,917 KB:**

Home page: <https://www.openssl-library.org/>

Download: <https://github.com/openssl/openssl/releases/download/openssl-3.4.1/openssl-3.4.1.tar.gz>

MD5 sum: fb7a747ac6793a7ad7118eaba45db379

- **Patch (2.7.6) - 766 KB:**

Home page: <https://savannah.gnu.org/projects/patch/>

Download: <https://ftp.gnu.org/gnu/patch/patch-2.7.6.tar.xz>

MD5 sum: 78ad9937e4caadcba1526ef1853730d5

- **Perl (5.40.1) - 13,605 KB:**

Home page: <https://www.perl.org/>

Download: <https://www.cpan.org/src/5.0/perl-5.40.1.tar.xz>

MD5 sum: bab3547a5cdf2302ee0396419d74a42e

- **Pkgconf (2.3.0) - 309 KB:**

Home page: <https://github.com/pkgconf/pkgconf>

Download: <https://distfiles.ariadne.space/pkgconf/pkgconf-2.3.0.tar.xz>

MD5 sum: 833363e77b5bed0131c7bc4cc6f7747b

- **Procps (4.0.5) - 1,483 KB:**

Home page: <https://gitlab.com/procps-ng/procps/>

Download: <https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-4.0.5.tar.xz>

MD5 sum: 90803e64f51f192f3325d25c3335d057

- **Psmisc (23.7) - 423 KB:**

Home page: <https://gitlab.com/psmisc/psmisc>

Download: <https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.7.tar.xz>

MD5 sum: 53eae841735189a896d614cba440eb10

- **Python (3.13.2) - 22,091 KB:**

Home page: <https://www.python.org/>

Download: <https://www.python.org/ftp/python/3.13.2/Python-3.13.2.tar.xz>

MD5 sum: 4c2d9202ab4db02c9d0999b14655dfe5

- **Python Documentation (3.13.2) - 10,102 KB:**

Download: <https://www.python.org/ftp/python/doc/3.13.2/python-3.13.2-docs-html.tar.bz2>

MD5 sum: d6aede88f480a018d26b3206f21654ae

- **Readline (8.2.13) - 2,974 KB:**

Home page: <https://tiswww.case.edu/php/chet/readline/r1top.html>

Download: <https://ftp.gnu.org/gnu/readline/readline-8.2.13.tar.gz>

MD5 sum: 05080bf3801e6874bb115cd6700b708f

- **Sed (4.9) - 1,365 KB:**

Home page: <https://www.gnu.org/software/sed/>

Download: <https://ftp.gnu.org/gnu/sed/sed-4.9.tar.xz>

MD5 sum: 6aac9b2dbafcd5b7a67a8a9bcb8036c3

- **SetupTools (75.8.1) - 1,313 KB:**

Home page: <https://pypi.org/project/setuptools/>

Download: <https://pypi.org/packages/source/s/setuptools/setuptools-75.8.1.tar.gz>

MD5 sum: 7dc3d3f529b76b10e35326e25c676b30

- **Shadow (4.17.3) - 2,274 KB:**

Home page: <https://github.com/shadow-maint/shadow/>

Download: <https://github.com/shadow-maint/shadow/releases/download/4.17.3/shadow-4.17.3.tar.xz>

MD5 sum: 0da190e53ecee76237e4c8f3f39531ed

- **Sysklogd (2.7.0) - 465 KB:**

Home page: <https://www.infodrom.org/projects/sysklogd/>

Download: <https://github.com/troglobit/sysklogd/releases/download/v2.7.0/sysklogd-2.7.0.tar.gz>

MD5 sum: 611c0fa5c138eb7a532f3c13bdf11ebc

- **Systemd (257.3) - 15,847 KB:**

Home page: <https://www.freedesktop.org/wiki/Software/systemd/>

Download: <https://github.com/systemd/systemd/archive/v257.3/systemd-257.3.tar.gz>

MD5 sum: 8e4fc90c7aead651fa5c50bd1b34abc2

- **Systemd Man Pages (257.3) - 733 KB:**

Home page: <https://www.freedesktop.org/wiki/Software/systemd/>

Download: <https://anduin.linuxfromscratch.org/LFS/systemd-man-pages-257.3.tar.xz>

MD5 sum: 9b77c3b066723d490cb10aed4fb05696



Note

Команда Linux From Scratch створює власний архів тап-сторінок, використовуючи вихідний код systemd. Це робиться для того, щоб уникнути непотрібних залежностей.

- **SysVinit (3.14) - 236 KB:**

Home page: <https://savannah.nongnu.org/projects/sysvinit>

Download: <https://github.com/slicer69/sysvinit/releases/download/3.14/sysvinit-3.14.tar.xz>

MD5 sum: bc6890b975d19dc9db42d0c7364dd092

- **Tar (1.35) - 2,263 KB:**

Home page: <https://www.gnu.org/software/tar/>

Download: <https://ftp.gnu.org/gnu/tar/tar-1.35.tar.xz>

MD5 sum: a2d8042658cf8ea939e6d911eaf4152

- **Tcl (8.6.16) - 11,406 KB:**

Home page: <https://tcl.sourceforge.net/>

Download: <https://downloads.sourceforge.net/tcl/tcl8.6.16-src.tar.gz>

MD5 sum: eaef5d0a27239fb840f04af8ec608242

- **Tcl Documentation (8.6.16) - 1,169 KB:**

Download: <https://downloads.sourceforge.net/tcl/tcl8.6.16-html.tar.gz>

MD5 sum: 750c221bcb6f8737a6791c1fbe98b684

- **Texinfo (7.2) - 6,259 KB:**

Home page: <https://www.gnu.org/software/texinfo/>

Download: <https://ftp.gnu.org/gnu/texinfo/texinfo-7.2.tar.xz>

MD5 sum: 11939a7624572814912a18e76c8d8972

- **Time Zone Data (2025a) - 453 KB:**

Home page: <https://www.iana.org/time-zones>

Download: <https://www.iana.org/time-zones/repository/releases/tzdata2025a.tar.gz>

MD5 sum: 404229390c06b7440f5e48d12c1a3251

- **Udev-lfs Tarball (udev-lfs-20230818) - 10 KB:**

Download: <https://anduin.linuxfromscratch.org/LFS/udev-lfs-20230818.tar.xz>

MD5 sum: acd4360d8a5c3ef320b9db88d275dae6

- **Util-linux (2.40.4) - 8,641 KB:**

Home page: [https://git.kernel.org/pub/scm/utils/util-linux/util-linux.git/](https://git.kernel.org/pub/scm/utils/util-linux/util-linux.git)

Download: <https://www.kernel.org/pub/linux/utils/util-linux/v2.40/util-linux-2.40.4.tar.xz>

MD5 sum: f9cbb1c8315d8ccbeb0ec36d10350304

- **Vim (9.1.1166) - 18,077 KB:**

Home page: <https://www.vim.org>

Download: <https://github.com/vim/vim/archive/v9.1.1166/vim-9.1.1166.tar.gz>

MD5 sum: 718d43ce957ab7c81071793de176c2eb



Note

Версія vim змінюється щодня. Щоб отримати останню версію, перейдіть на сайт <https://github.com/vim/vim/tags>.

- **Wheel (0.45.1) - 106 KB:**

Home page: <https://pypi.org/project/wheel/>

Download: <https://pypi.org/packages/source/w/wheel/wheel-0.45.1.tar.gz>

MD5 sum: dddc505d0573d03576c7c6c5a4fe0641

- **XML::Parser (2.47) - 276 KB:**

Home page: <https://github.com/chorny/XML-Parser>

Download: <https://cpn.metacpan.org/authors/id/T/TO/TODDR/XML-Parser-2.47.tar.gz>

MD5 sum: 89a8e82cf2ad948b349c0a69c494463

- **Xz Utils (5.6.4) - 1,310 KB:**

Home page: <https://tukaani.org/xz>

Download: <https://github.com/tukaani-project/xz/releases/download/v5.6.4/xz-5.6.4.tar.xz>

MD5 sum: 4b1cf07d45ec7eb90a01dd3c00311a3e

- **Zlib (1.3.1) - 1,478 KB:**

Home page: <https://zlib.net/>

Download: <https://zlib.net/fossils/zlib-1.3.1.tar.gz>

MD5 sum: 9855b6d802d7fe5b7bd5b196a2271655

- **Zstd (1.5.7) - 2,378 KB:**

Home page: <https://facebook.github.io/zstd/>

Download: <https://github.com/facebook/zstd/releases/download/v1.5.7/zstd-1.5.7.tar.gz>

MD5 sum: 780fc1896922b1bc52a4e90980cdda48

Загальний розмір цих пакетів: приблизно 527 МБ

3.3. Необхідні патчі

Окрім пакетів, також необхідні декілька патчів. Ці патчі виправляють помилки в пакетах, які повинні бути виправлені розробником. Патчі також вносять невеликі зміни, щоб полегшити роботу з пакетами. Для побудови системи LFS необхідні наступні патчі:

- **Bzip2 Documentation Patch - 1.6 KB:**

Download: https://www.linuxfromscratch.org/patches/lfs/12.3/bzip2-1.0.8-install_docs-1.patch

MD5 sum: 6a5ac7e89b791aae556de0f745916f7f

- **Coreutils Internationalization Fixes Patch - 164 KB:**

Download: <https://www.linuxfromscratch.org/patches/lfs/12.3/coreutils-9.6-i18n-1.patch>

MD5 sum: 6aee45dd3e05b7658971c321d92f44b7

- **Expect GCC14 Patch - 7.8 KB:**

Download: <https://www.linuxfromscratch.org/patches/lfs/12.3/expect-5.45.4-gcc14-1.patch>

MD5 sum: 0b8b5ac411d011263ad40b0664c669f0

- **Glibc FHS Patch - 2.8 KB:**

Download: <https://www.linuxfromscratch.org/patches/lfs/12.3/glibc-2.41-fhs-1.patch>

MD5 sum: 9a5997c3452909b1769918c759eff8a2

- **Kbd Backspace/Delete Fix Patch - 12 KB:**

Download: <https://www.linuxfromscratch.org/patches/lfs/12.3/kbd-2.7.1-backspace-1.patch>

MD5 sum: f75cca16a38da6caa7d52151f7136895

- **SysVinit Consolidated Patch - 2.5 KB:**

Download: <https://www.linuxfromscratch.org/patches/lfs/12.3/sysvinit-3.14-consolidated-1.patch>

MD5 sum: 3af8fd8e13cad481eeeeaa48be4247445

Загальний розмір цих патчів: приблизно 190,7 КБ

Окрім вищезазначених необхідних патчів, існує низка додаткових патчів, створених спільногою LFS. Ці додаткові патчі вирішують дрібні проблеми або вмикають функції, які за замовчуванням не ввімкнені. Ви можете переглянути базу даних патчів за адресою <https://www.linuxfromscratch.org/patches/downloads/> і завантажити додаткові патчі, які відповідають потребам вашої системи.

4. Фінальні приготування

4.1. Вступ

У цьому розділі ми виконаємо кілька додаткових завдань, щоб підготуватися до побудови тимчасової системи. Ми створимо набір каталогів у \$LFS (в яких ми встановимо тимчасові інструменти), додамо користувача без привileїв і створимо відповідне середовище побудови для цього користувача. Ми також пояснимо одиниці часу («SBU»), які ми використовуємо для вимірювання часу, необхідного для побудови пакетів LFS, і надамо деяку інформацію про набори тестів пакетів.

4.2. Створення обмеженої структури каталогів у файловій системі LFS

У цьому розділі ми починаємо заповнювати файлову систему LFS елементами, які складуть остаточну систему Linux. Першим кроком є створення обмеженої ієрархії каталогів, щоб програми, скомпільовані в розділі 6 (а також glibc і libstdc++ в розділі 5), могли бути встановлені в їх остаточне місце розташування. Ми робимо це для того, щоб ці тимчасові програми були перезаписані, коли остаточні версії будуть побудовані в розділі 8.

Створіть необхідну структуру каталогів, виконавши наступні команди як `root`:

```
mkdir -pv $LFS/{etc,var} $LFS/usr/{bin,lib,sbin}

for i in bin lib sbin; do
    ln -sv usr/$i $LFS/$i
done

case $(uname -m) in
    x86_64) mkdir -pv $LFS/lib64 ;;
esac
```

Програми в розділі 6 будуть компілюватися за допомогою крос-компілятора (більш детальну інформацію можна знайти в розділі Технічні нотатки інструментарію). Цей крос-компілятор буде встановлено в спеціальному каталозі, щоб відокремити його від інших програм. Все ще діючи як `root`, створіть цей каталог за допомогою такої команди:

```
mkdir -pv $LFS/tools
```



Note

Версія vim змінюється щодня. Щоб отримати останню версію, перейдіть на сайт <https://github.com/vim/vim/tags>.

Редактори LFS свідомо вирішили не використовувати каталог `/usr/lib64`. Було вжито кілька заходів, щоб переконатися, що набір інструментів не буде його використовувати. Якщо з якоїсь причини цей каталог з'явиться (або через те, що ви помилилися під час виконання інструкцій, або через те, що ви встановили

бінарний пакет, який створив його після завершення LFS), це може пошкодити вашу систему. Ви завжди повинні переконатися, що цей каталог не існує.

4.3. Додавання користувача LFS

Коли ви ввійшли в систему як користувач `root`, одна помилка може пошкодити або знищити систему. Тому пакети в наступних двох розділах створюються від імені непривілейованого користувача. Ви можете використовувати своє власне ім'я користувача, але щоб полегшити налаштування чистого робочого середовища, ми створимо нового користувача з іменем `lfs` як члена нової групи (також з іменем `lfs`) і виконуватимемо команди як `lfs` під час процесу інсталяції. Як `root`, виконайте наступні команди, щоб додати нового користувача:

```
groupadd lfs
useradd -s /bin/bash -g lfs -m -k /dev/null lfs
```

Ось що означають параметри командного рядка:

`-s /bin/bash`

Це робить `bash` оболонкою за замовчуванням для користувача `lfs`.

`-g lfs`

Ця опція додає користувача `lfs` до групи `lfs`.

`-m`

Це створює домашній каталог для `lfs`.

`-k /dev/null`

Цей параметр запобігає можливому копіюванню файлів із каталогу-скелета (за замовчуванням це `/etc/skel`), змінюючи місце введення на спеціальний нульовий пристрій.

`lfs`

Це ім'я нового користувача.

Якщо ви хочете увійти в систему як `lfs` або перейти до `lfs` з некореневого користувача (на відміну від переходу до користувача `lfs` під час входу в систему як `root`, що не вимагає від користувача `lfs` мати пароль), вам потрібно встановити пароль для `lfs`. Виконайте наступну команду як користувач `root`, щоб встановити пароль:

```
passwd lfs
```

Надайте `lfs` повний доступ до всіх каталогів під `$LFS`, зробивши `lfs` власником:

```
chown -v lfs $LFS/{usr{,/*},var,etc,tools}
case $(uname -m) in
  x86_64) chown -v lfs $LFS/lib64 ;;
esac
```

**Note**

У деяких хост-системах наступна команда **su** не виконується належним чином і призупиняє вхід користувача **lfs** у фоновому режимі. Якщо підказка «**lfs:~\$**» не з'являється відразу, введення команди **fg** вирішить цю проблему.

Далі запустіть оболонку, що працює від імені користувача **lfs**. Це можна зробити, увійшовши в систему як **lfs** на віртуальній консолі або за допомогою наступної команди заміни/перемікання користувача:

```
su - lfs
```

Знак «**--**» вказує **su** запустити оболонку для входу в систему, а не оболонку без входу в систему. Різниця між цими двома типами оболонок детально описана в *bash(1)* та **info bash**.

4.4. Налаштування оточення

Створіть хороше робоче середовище, створивши два нових файли запуску для оболонки **bash**. Увійшовши в систему як користувач **lfs**, виконайте наступну команду, щоб створити новий файл **.bash_profile**:

```
cat > ~/.bash_profile << "EOF"
exec env -i HOME=$HOME TERM=$TERM PS1='\u:\w\$ ' /bin/bash
EOF
```

Коли ви ввійшли в систему як користувач **lfs** або перейшли до користувача **lfs** за допомогою команди **su** з опцією «**--**», початковою оболонкою є оболонка входу, яка читає файл **/etc/profile** хоста (який, ймовірно, містить деякі налаштування та змінні середовища), а потім файл **.bash_profile**. Команда **exec env -i.../bin/bash** у файлі **.bash_profile** замінює запущену оболонку на нову з повністю порожнім середовищем, за винятком змінних **HOME**, **TERM** і **PS1**. Це гарантує, що жодні небажані та потенційно небезпечні змінні середовища з хост-системи не потраплять у середовище побудови.

Новий екземпляр оболонки є оболонкою без входу в систему, яка не читає і не виконує вміст файлів **/etc/profile** або **.bash_profile**, а замість цього читає і виконує файл **.bashrc**. Створіть файл **.bashrc** зараз:

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
LFS_TGT=$(uname -m)-lfs-linux-gnu
PATH=/usr/bin
if [ ! -L /bin ]; then PATH=/bin:$PATH; fi
PATH=$LFS/tools/bin:$PATH
CONFIG_SITE=$LFS/usr/share/config.site
export LFS LC_ALL LFS_TGT PATH CONFIG_SITE
EOF
```

Значення налаштувань у файлі .bashrc

```
set +h
```

Команда **set +h** вимикає хеш-функцію **bash**. Хешування зазвичай є корисною функцією — **bash** використовує хеш-таблицю для запам'ятовування повного шляху до виконуваних файлів, щоб уникнути багаторазового пошуку в PATH того самого виконуваного файла. Однак нові інструменти слід використовувати одразу після їх встановлення. Вимкнення хеш-функції змушує оболонку шукати в PATH щоразу, коли потрібно запустити програму. Таким чином, оболонка знайде новоскомпільовані інструменти в `$LFS/tools/bin`, як тільки вони стануть доступними, не запам'ятуючи попередню версію тієї ж програми, надану дистрибутивом хоста, в `/usr/bin` або `/bin`.

```
umask 022
```

Встановлення `umask`, як ми вже пояснювали в розділі 2.6, «Встановлення змінної \$LFS і `umask`».

```
LFS=/mnt/lfs
```

Змінна `LFS` повинна бути встановлена на вибрану точку монтування.

```
LC_ALL=POSIX
```

Змінна `LC_ALL` контролює локалізацію певних програм, змушуючи їхні повідомлення відповідати конвенціям визначеній країни. Встановлення `LC_ALL` на «POSIX» або «C» (ці два значення є еквівалентними) гарантує, що все буде працювати як очікується в середовищі крос-компіляції.

```
LFS_TGT=$(uname -m)-lfs-linux-gnu
```

Змінна `LFS_TGT` встановлює нестандартний, але сумісний опис машини для використання під час побудови нашого крос-компілятора та лінкера, а також під час крос-компіляції нашого тимчасового набору інструментів. Більш детальна інформація наведена в Технічних нотатках інструментарію.

```
PATH=/usr/bin
```

Багато сучасних дистрибутивів Linux об'єднали `/bin` та `/usr/bin`. У цьому випадку стандартна змінна `PATH` повинна бути встановлена на `/usr/bin/` для середовища розділу 6. Якщо це не так, наступний рядок додає `/bin` до шляху.

```
if [ ! -L /bin ]; then PATH=/bin:$PATH; fi
```

Якщо `/bin` не є символічним посиланням, його потрібно додати до змінної `PATH`.

```
PATH=$LFS/tools/bin:$PATH
```

Помістивши `$LFS/tools/bin` перед стандартним `PATH`, крос-компілятор, встановлений на початку розділу 5, відразу після встановлення буде підхоплений оболонкою. Це, в поєднанні з вимкненням хешування, обмежує ризик використання компілятора з хоста замість крос-компілятора.

```
CONFIG_SITE=$LFS/usr/share/config.site
```

У розділах 5 і 6, якщо ця змінна не встановлена, скрипти `configure` можуть спробувати завантажити елементи конфігурації, характерні для деяких дистрибутивів, з `/usr/share/config.site` на хост-системі. Перезапишіть її, щоб запобігти потенційному зараженню від хоста.

```
export ...
```

Хоча попередні команди встановили деякі змінні, щоб зробити їх видимими в будь-яких підоболонках, ми експортуємо їх.



Important

Декілька комерційних дистрибутивів додають недокументовану інстанцію `/etc/bash.bashrc` до ініціалізації `bash`. Цей файл може змінити середовище користувача `lfs` таким чином, що це може вплинути на побудову критичних пакетів LFS. Щоб переконатися, що середовище користувача `lfs` є чистим, перевірте наявність `/etc/ bash.bashrc` і, якщо він є, перемістіть його в інше місце. Як користувач `root`, виконайте:



Important

```
[ ! -e /etc/bash.bashrc ] || mv -v /etc/bash.bashrc /etc/bash.bashrc.NOUSE
```

Коли користувач `lfs` більше не потрібен (на початку розділу 7), ви можете безпечно відновити `/etc/bash.bashrc` (за бажанням).

Зверніть увагу, що пакет LFS Bash, який ми будемо створювати в розділі 8.36, «Bash-5.2.37», не налаштований для завантаження або виконання `/etc/bash.bashrc`, тому цей файл є непотрібним у готовій системі LFS.

Для багатьох сучасних систем з декількома процесорами (або ядрами) час компіляції пакета можна скоротити, виконавши «паралельне створення», вказавши програмі `make`, скільки процесорів доступно, за допомогою опції командного рядка або змінної середовища. Наприклад, процесор Intel Core i9-13900K має 8 Р (продуктивних) ядер і 16 Е (ефективних) ядер, а Р-ядро може одночасно виконувати два потоки, тому кожне Р-ядро моделюється як два логічних ядра ядром Linux. В результаті загалом є 32 логічних ядра. Очевидний спосіб використання всіх цих логічних ядер — дозволити `make` запускати до 32 завдань компіляції. Це можна зробити, передавши `make` опцію `-j32`:

```
make -j32
```

Або встановіть змінну середовища `MAKEFLAGS`, і її вміст буде автоматично використано `make` як параметри командного рядка.

```
export MAKEFLAGS=-j32
```



Important

Ніколи не передавайте опцію `-j` без числа для `make` або не встановлюйте таку опцію в `MAKEFLAGS`. Це дозволить `make` запускати нескінченні завдання побудови і спричинить проблеми зі стабільністю системи.

Щоб використовувати всі логічні ядра, доступні для створення пакетів у розділах 5 і 6, встановіть `MAKEFLAGS` зараз у `.bashrc`:

```
cat >> ~/.bashrc << "EOF"
```

```
export MAKEFLAGS=-j$(nproc)
EOF
```

Замініть `$(nproc)` на кількість логічних ядер, які ви хочете використовувати, якщо не хочете використовувати всі логічні ядра.

Нарешті, щоб переконатися, що середовище повністю готове до створення тимчасових інструментів, змусьте оболонку **bash** прочитати новий профіль користувача:

```
source ~/.bash_profile
```

4.5. Про SBUs

Багато людей хотіли б заздалегідь знати, скільки часу приблизно займає компіляція та встановлення кожного пакета. Оскільки Linux From Scratch можна побудувати на багатьох різних системах, неможливо надати абсолютні оцінки часу. Найбільший пакет (gcc) займе приблизно 5 хвилин на найшвидших системах, але може зайняти кілька днів на повільніших системах! Замість надання фактичного часу, буде використовуватися міра Standard Build Unit (SBU).

Mіра SBU працює наступним чином. Першим пакетом, який компілюється, є `binutils` у розділі 5. Час, який потрібно для компіляції з використанням одного ядра, ми будемо називати стандартною одиницею компіляції або SBU. Усі інші часи компіляції будуть виражатися в цій одиниці часу.

Наприклад, розглянемо пакет, час компіляції якого становить 4,5 SBU. Це означає, що якщо ваша система витратила 4 хвилини на компіляцію та встановлення первого проходу `binutils`, то на побудову прикладу пакета знадобиться приблизно 18 хвилин. На щастя, більшість часів компіляції коротші за один SBU.

SBU не є цілком точними, оскільки залежать від багатьох факторів, включаючи версію GCC в хост-системі. Вони надаються тут для того, щоб дати приблизну оцінку часу, який може знадобитися для встановлення пакета, але цифри можуть відрізнятися на десятки хвилин у деяких випадках.

У деяких новіших системах материнська плата здатна контролювати тактову частоту системи. Це можна контролювати за допомогою команди, такої як **powerprofilesctl**. Ця функція недоступна в LFS, але може бути доступна в дистрибутиві хоста. Після завершення LFS його можна додати до системи за допомогою процедур, описаних на сторінці BLFS `power-profiles-daemon`. Перед вимірюванням часу побудови будь-якого пакета рекомендується використовувати профіль живлення системи, налаштований на максимальну продуктивність (і максимальне споживання енергії). В іншому випадку виміряне значення SBU може бути неточним, оскільки система може реагувати по-різному під час компіляції `binutils` – етапу 1 або інших пакетів. Майте на увазі, що значна неточність може все одно проявитися навіть якщо для обох пакетів використовується один і той же профіль, оскільки система може реагувати повільніше, якщо вона перебуває в режимі очікування під час запуску процедури компіляції. Встановлення профілю живлення на «`performance`» мінімізує цю проблему. I, очевидно, це також прискорить компіляцію LFS системою.

Якщо доступна команда **powerprofilesctl**, виконайте команду **powerprofilesctl set performance**, щоб вибрати профіль продуктивності. Деякі дистрибутиви надають команду `tuned-adm` для управління

профілями замість **powerprofilesctl**. У цих дистрибутивах виконайте команду **tuned-adm profile throughput-performance**, щоб вибрати профіль пропускної здатності та продуктивності.



Note

Коли використовується кілька процесорів, одиниці SBU в кнізі будуть відрізнятися ще більше, ніж зазвичай. У деяких випадках крок `make` просто не вдається. Аналіз результатів процесу побудови також буде складнішим, оскільки рядки з різних процесів будуть перемежковуватися. Якщо ви зіткнулися з проблемою на етапі побудови, поверніться до побудови на одному процесорі, щоб правильно проаналізувати повідомлення про помилки.

Час, представлений тут для всіх пакетів (крім `binutils-pass1`, який базується на одному ядрі), базується на використанні чотирьох ядер (`-j4`). Час у розділі 8 також включає час на виконання регресійних тестів для пакета, якщо не вказано інше.

4.6. Про набори тестів

Більшість пакетів містять набір тестів. Виконання набору тестів для новоствореного пакета є гарною ідеєю, оскільки це дозволяє перевірити, чи все скомпільовано правильно. Набір тестів, який проходить перевірку, зазвичай доводить, що пакет функціонує так, як передбачав розробник. Однак це не гарантує, що пакет є повністю вільним від помилок.

Деякі набори тестів є важливішими за інші. Наприклад, набори тестів для основних пакетів інструментарію — `GCC`, `binutils` та `glibc` — мають надзвичайне значення через їхню центральну роль у належному функціонуванні системи. Виконання наборів тестів для `GCC` та `glibc` може зайняти дуже багато часу, особливо на повільному обладнанні, але їх настійно рекомендується виконувати.



Note

Виконання наборів тестів у розділах 5 і 6 не має сенсу, оскільки тестові програми компілюються за допомогою крос-компілятора, тому вони, ймовірно, не зможуть працювати на хості збірки.

Пошироною проблемою при виконанні наборів тестів для `binutils` та `GCC` є вичерпання псевдотерміналів (PTY). Це може привести до великої кількості невдалих тестів. Це може статися з кількох причин, але найімовірнішою причиною є те, що в хост-системі неправильно налаштована файлова система `devpts`. Ця проблема більш детально розглядається на сторінці <https://www.linuxfromscratch.org/lfs/faq.html#no-ptys>.

Іноді набори тестів пакетів провалюються з причин, про які розробники знають і які вони вважають некритичними. Зверніться до журналів, розташованих за адресою <https://www.linuxfromscratch.org/lfs/build-logs/12.3/>, щоб перевірити, чи є ці провали очікуваними. Цей сайт є дійсним для всіх наборів тестів у цій книзі.

Частина III. Створення крос-компілятора LFS та тимчасових інструментів

Важливі попередні матеріали

Вступ

Ця частина поділяється на три етапи: по-перше, створення крос-компілятора та пов'язаних з ним бібліотек; по-друге, використання цього крос-інструментарію для створення декількох утиліт таким чином, щоб ізолювати їх від дистрибутива хоста; і по-третє, вхід в середовище `chroot` (що ще більше покращує ізоляцію хоста) та створення решти інструментів, необхідних для побудови кінцевої системи.



Important

Тут починається справжня робота з побудови нової системи. Будьте дуже уважні і дотримуйтесь інструкцій точно так, як вони наведені в книзі. Ви повинні намагатися зрозуміти, що робить кожна команда, і незалежно від того, наскільки ви прагнете закінчити побудову, не слід сліпо вводити команди, як показано. Прочитайте документацію, якщо є щось, чого ви не розумієте. Також стежте за тим, що ви вводите, і за результатами виконання команд, використовуючи утиліту `tee` для відправки виводу терміналу в файл. Це полегшить налагодження, якщо щось піде не так.

Наступний розділ — це технічний вступ до процесу збірки, а потім йде розділ із **дуже важливими** загальними інструкціями.

Технічні нотатки інструментарію

Цей розділ пояснює деякі технічні деталі та обґрунтування загального методу збірки. Не намагайтесь відразу зрозуміти все в цьому розділі. Більшість цієї інформації стане зрозумілішою після виконання фактичної збірки. Поверніться і перечитайте цей розділ у будь-який час під час процесу збірки.

Основна мета Розділу 5 і Розділу 6 — створити тимчасову область, яка міститиме набір перевірених інструментів, ізольованих від основної системи. За допомогою команди `chroot` компіляція в решті розділів буде ізольована в цьому середовищі, що забезпечить чисту, безпроблемну збірку цільової системи LFS. Процес збірки був розроблений з метою мінімізації ризиків для нових читачів і надання максимальної освітньої цінності одночасно.

Цей процес збірки базується на крос-компіляції. Перехресна компіляція зазвичай використовується для побудови компілятора та пов'язаного з ним набору інструментів для машини, відмінної від тієї, що використовується для збірки. Це не є строго необхідним для LFS, оскільки машина, на якій буде працювати нова система, є такою ж, як і та, що використовується для побудови. Але крос-компіляція має одну велику перевагу: все, що перехресно компілюється, не може залежати від хост-середовища.

Про крос-компіляцію



Note

Книга LFS не є (і не містить) загальним посібником з побудови крос- (або нативного) інструментарію. Не використовуйте команди з книги для крос-інструментарію з якоюсь іншою метою, окрім побудови LFS, якщо ви не розумієте, що саме робите.

Крос-компіляція включає в себе деякі поняття, які заслуговують на окремий розділ. Хоча цей розділ можна пропустити при першому прочитанні, повернення до нього пізніше допоможе вам отримати більш повне розуміння процесу.

Спочатку давайте визначимо деякі терміни, що використовуються в цьому контексті.

Машина для збірки (the build)

це машина, на якій ми складаємо програми. Зверніть увагу, що ця машина також називається «хостом».

Хост (the host)

це машина/система, на якій будуть працювати складені програми. Зверніть увагу, що це використання терміна «хост» не є таким самим, як в інших розділах.

Ціль (the target)

використовується тільки для компіляторів. Це машина, для якої компілятор створює код. Вона може відрізнятися як від складання, так і від хосту.

Як приклад, уявімо собі такий сценарій (іноді його називають «канадським хрестом»). Ми маємо компілятор тільки на повільній машині, назвемо її машиною А, і компілятор ccA. Ми також маємо швидку машину (В), але не маємо компілятора для (В), і ми хочемо створити код для третьої, повільної машини (С). Ми будемо створювати компілятор для машини С у три етапи.

Етап	Збірка	Хост	Ціль	Дія
1	A	A	B	Скомпілюйте крос-компілятор cc1 за допомогою ccA на машині A.
2	A	B	C	Створити крос-компілятор cc2 за допомогою cc1 на машині A.
3	B	C	C	Скомпілюйте компілятор ccC за допомогою cc2 на машині B.

Тоді всі програми, необхідні для машини С, можна скомпілювати за допомогою cc2 на швидкій машині В. Зверніть увагу, що якщо В не може запускати програми, створені для С, тому немає можливості перевірити новостворені програми, поки сама машина С не запрацює. Наприклад, щоб запустити набір тестів на ccC, ми можемо додати четвертий етап:

Етап	Збірка	Хост	Ціль	Дія
4	C	C	C	Перебудувати і протестувати ccC, використовуючи ccC на машині C.

У наведеному вище прикладі тільки cc1 і cc2 є крос-компіляторами, тобто вони генерують код для машини, відмінної від тієї, на якій вони працюють. Інші компілятори ccA і ccC генерують код для машини, на якій вони працюють. Такі компілятори називаються нативними компіляторами.

Реалізація крос-компіляції для LFS

Note

Усі крос-компільовані пакети в цій книзі використовують систему побудови на основі autoconf. Система побудови на основі autoconf приймає типи систем у формі `cpu-vendor-kernel-os`, що називається системним тріплетом. Оскільки поле vendor часто не має значення, autoconf дозволяє його опустити.

Проникливий читач може запитати, чому «трійка» позначає назву, що складається з чотирьох компонентів. Поле kernel і поле os спочатку були одним полем «system». Така форма з трьома полями досі є дійсною для деяких систем, наприклад, `x86_64-unknown-freebsd`. Але дві системи можуть мати однакове ядро і все одно бути надто різними, щоб використовувати одну і ту ж трійку для їх опису. Наприклад, Android, що працює на мобільному телефоні, повністю відрізняється від Ubuntu, що працює на сервері ARM64, хоча обидві системи працюють на одному типі процесора (ARM64) і використовують одне ядро (Linux).

Без рівня емуляції ви не можете запустити виконуваний файл для сервера на мобільному телефоні або навпаки. Тому поле «system» було розділено на поля kernel та os, щоб однозначно позначити ці системи. У нашому прикладі система Android позначена як `aarch64-unknown-linux-android`, а система Ubuntu — як `aarch64-unknown-linux-gnu`.

Слово «трійка» залишається в лексиконі. Простий спосіб визначити трійку вашої системи — запустити скрипт `config.guess`, який входить до складу багатьох пакетів. Розпакуйте джерела binutils, запустіть скрипт `./config.guess` і запишіть результат. Наприклад, для 32-бітного процесора Intel результатом буде `i686-pc-linux-gnu`. На 64-бітній системі це буде `x86_64-pc-linux-gnu`. На більшості систем Linux ще простіша команда `gcc-dumpmachine` надасть вам подібну інформацію.

Ви також повинні знати назву динамічного лінкера платформи, який часто називають динамічним завантажувачем (не плутати зі стандартним лінкером `ld`, що входить до складу binutils). Динамічний лінкер, що надається пакетом glibc, знаходить і завантажує спільні бібліотеки, необхідні програмі, готує програму до запуску, а потім запускає її. Назва динамічного лінкера для 32-бітної машини Intel — `ld-linux.so.2`; для 64-бітної машини — `ld-linux-x86-64.so.2` на 64-бітних системах. Надійний спосіб визначити ім'я динамічного лінкера — перевірити випадковий бінарний файл з хост-системи, виконавши: `readelf -l <ім'я бінарного файлу> | grep interpreter` і записавши результат. Авторитетний довідковий матеріал, що охоплює всі платформи, знаходиться на сторінці [Glibc wiki](#).

Щоб імітувати крос-компіляцію в LFS, ім'я трійки хоста дещо коригується шляхом зміни поля «vendor» у змінній `LFS_TGT`, щоб воно містило «lfs». Ми також використовуємо опцію `--with-sysroot` під час побудови крос-лінкера та крос-компілятора, щоб вказати їм, де знайти необхідні файли хоста. Це гарантує, що жодна з інших програм, побудованих у розділі 6, не зможе зв'язатися з бібліотеками на машині побудови. Обов'язковими є лише два етапи, плюс ще один для тестування.

Етап	Збірка	Хост	Ціль	Дія
1	pc	pc	lfs	Зібрання крос-компілятора cc1 за допомогою cc-lfs на машині pc.
2	pc	lfs	lfs	Створити крос-компілятор cc-lfs за допомогою cc1 на машині pc.
3	lfs	lfs	lfs	Перебудувати і протестувати cc-lfs, використовуючи cc-lfs на машині lfs.

У попередній таблиці «на машині pc» означає, що команди виконуються на машині з уже встановленим дистрибутивом. «на машині lfs» означає, що команди виконуються в середовищі chroot.

Це ще не кінець історії. Мова С — це не просто компілятор, вона також визначає стандартну бібліотеку. У цій книзі використовується бібліотека GNU C під назвою glibc (існує також альтернатива — «musl»). Ця бібліотека повинна бути скомпільована для машини LFS, тобто за допомогою крос-компілятора cc1. Але сам компілятор використовує внутрішню бібліотеку, що надає складні підпрограми для функцій, які не доступні в наборі інструкцій асемблера. Ця внутрішня бібліотека називається libgcc, і вона повинна бути пов'язана з бібліотекою glibc, щоб бути повністю функціональною. Крім того, стандартна бібліотека для C++ (libstdc++) також повинна бути пов'язана з glibc. Вирішенням цієї проблеми «яйця і курки» є спочатку побудувати погіршенню libgcc на основі cc1, без деяких функціональних можливостей, таких як потоки і обробка винятків, а потім побудувати glibc, використовуючи цей погіршений компілятор (glibc сама по собі не погіршена), а також побудувати libstdc++. Ця остання бібліотека не матиме деяких функціональних можливостей libgcc.

Висновок з попереднього абзацу полягає в тому, що cc1 не може створити повністю функціональну libstdc++ з погіршеною libgcc, але cc1 є єдиним компілятором, доступним для створення бібліотек C/C++ на етапі 2. Є дві причини, чому ми не використовуємо компілятор, створений на етапі 2, cc-lfs, для створення цих бібліотек.

- Загалом кажучи, cc-lfs не може працювати на pc (гостевій системі). Навіть якщо трійки для pc і lfs сумісні між собою, виконуваний файл для lfs повинен залежати від glibc-2.41; гостевий дистрибутив може використовувати або іншу реалізацію libc (наприклад, musl), або попередню версію glibc (наприклад, glibc-2.13).
- Навіть якщо cc-lfs може працювати на ПК, його використання на ПК створює ризик зв'язку з бібліотеками ПК, оскільки cc-lfs є нативним компілятором.

Отже, коли ми будуємо gcc етап 2, ми даємо вказівку системі побудови перебудувати libgcc і libstdc++ за допомогою cc1, але ми пов'язуємо libstdc++ з новою перебудованою libgcc замість старої, погрішеної побудови. Це робить перебудовану libstdc++ повністю функціональною.

У розділі 8 (або «етапі 3») збираються всі пакети, необхідні для системи LFS. Навіть якщо пакет вже був встановлений в систему LFS в попередньому розділі, ми все одно перекомпілюємо цей пакет. Основною причиною перекомпіляції цих пакетів є забезпечення їх стабільності: якщо ми переінсталюємо пакет LFS у готову систему LFS, переінстальований вміст пакета повинен бути таким самим, як і вміст того самого пакета, що був вперше інстальований у розділі 8. Тимчасові пакети, встановлені в розділі 6 або розділі 7, не можуть задовольнити цю вимогу, оскільки деякі з них побудовані без опціональних залежностей, а autoconf не може виконати деякі перевірки функцій у розділі 6 через крос-компіляцію, що призводить до того, що тимчасові пакети не мають опціональних функцій або використовують неоптимальні кодові процедури. Крім того, другорядною причиною перекомпіляції пакетів є виконання наборів тестів.

Інші деталі процедур

Крос-компілятор буде встановлено в окремому каталозі `$LFS/tools`, оскільки він не буде частиною кінцевої системи.

Binutils встановлюється першим, оскільки **конфігурація** gcc і glibc виконує різні тести функціональних можливостей асемблера і лінкера, щоб визначити, які програмні функції слід увімкнути або вимкнути. Це важливіше, ніж може здатися на перший погляд. Неправильне налаштовані gcc або glibc можуть привести до незначного порушення роботи ланцюжка інструментів, причому наслідки такого порушення можуть проявитися лише наприкінці процесу побудови всього дистрибутива. Невдале виконання набору тестів зазвичай виявляє цю помилку до того, як буде виконано занадто багато додаткової роботи.

Binutils встановлює свій асемблер і лінкер у двох місцях: `$LFS/tools/bin` і `$LFS/tools/$LFS_TGT/bin`. Інструменти в одному місці жорстко пов'язані з іншим. Важливим аспектом лінкера є порядок пошуку бібліотек. Детальну інформацію можна отримати від `ld`, передавши йому прапор `--verbose`. Наприклад, `$LFS_TGT-ld --verbose | grep SEARCH` проілюструє поточні шляхи пошуку та їх порядок. (Зверніть увагу, що цей приклад можна виконати, як показано, тільки під час входу в систему як користувач `lfs`. Якщо ви повернетесь на цю сторінку пізніше, замініть `$LFS_TGT-ld` на `ld`).

Наступним встановленням пакетом є gcc. Приклад того, що можна побачити під час його запуску **configure**:

```
checking what assembler to use... /mnt/lfs/tools/i686-lfs-linux-gnu/bin/as
checking what linker to use... /mnt/lfs/tools/i686-lfs-linux-gnu/bin/ld
```

Це важливо з вищезазначених причин. Це також демонструє, що скрипт конфігурації gcc не шукає в каталогах PATH, які інструменти використовувати. Однак під час фактичної роботи самого gcc ті самі шляхи пошуку не обов'язково використовуються. Щоб дізнатися, який стандартний лінкер буде використовувати **gcc**, виконайте: `$LFS_TGT-gcc -print-prog-name=ld`. (Знову ж таки, видаліть префікс `$LFS_TGT-`, якщо повернетесь до цього пізніше.)

Детальну інформацію можна отримати від gcc, передавши йому опцію командного рядка `-v` під час компіляції програми. Наприклад, `$LFS_TGT-gcc -v example.c` (або без `$LFS_TGT-`, якщо повернетесь пізніше) покаже детальну інформацію про етапи препроцесора, компіляції та асемблювання, включаючи шляхи пошуку gcc для включених заголовків та їх порядок.

Далі: санізовані заголовки API Linux. Вони дозволяють стандартній бібліотеці C (glibc) взаємодіяти з функціями, які надає ядро Linux.

Далі йде glibc. Найважливішими факторами для побудови glibc є компілятор, бінарні інструменти та заголовки ядра. Компілятор та бінарні інструменти зазвичай не становлять проблеми, оскільки glibc завжди використовує ті, що пов'язані з параметром `--host`, переданим до його скрипту конфігурації; наприклад, у нашому випадку компілятором буде `$LFS_TGT-gcc`, а інструментом `readelf` буде `$LFS_TGT-readelf`. Заголовки ядра можуть бути дещо складнішими. Тому ми не ризикуємо і використовуємо доступний перемикач `configure`, щоб забезпечити правильний вибір. Після запуску `configure` перевірте вміст файлу `config.make` в каталозі збірки на наявність усіх важливих деталей. Ці елементи підкреслюють важливий аспект пакета glibc — він є дуже самодостатнім з точки зору механізму збірки і, як правило, не покладається на стандартні налаштування набору інструментів.

Як згадувалося вище, наступним кроком є компіляція стандартної бібліотеки C++, а в розділі 6 — інших програм, які необхідно компілювати в крос-середовищі, щоб уникнути циклічних залежностей під час побудови. Під час встановлення всіх цих пакетів використовується змінна `DESTDIR`, щоб примусово встановити їх у файлову систему LFS.

В кінці розділу 6 встановлюється нативний компілятор LFS. Спочатку будується binutils-етап2, в тому ж каталозі `DESTDIR`, що й інші програми, потім будується другий прохід gcc, оминаючи деякі некритичні бібліотеки. Через деяку дивну логіку в скрипті конфігурації gcc, `CC_FOR_TARGET` стає `CC`, коли хост є таким самим, як і ціль, але відрізняється від системи побудови. Ось чому `CC_FOR_TARGET=$LFS_TGT-gcc` явно оголошується як одна з опцій конфігурації.

Після входу в середовище chroot у розділі 7 виконується тимчасове встановлення програм, необхідних для належної роботи набору інструментів. З цього моменту основний набір інструментів є самодостатнім і самохостованим. У розділі 8 створюються, тестуються та встановлюються остаточні версії всіх пакетів, необхідних для повноцінної роботи системи.

Загальні інструкції компіляції



Caution

Під час циклу розробки LFS інструкції в книзі часто змінюються, щоб адаптуватися до оновлення пакета або скористатися новими функціями оновлених пакетів. Змішування інструкцій різних версій книги LFS може привести до незначних порушень. Такі проблеми зазвичай виникають внаслідок повторного використання деяких скриптів, створених для попередньої версії LFS. Таке повторне використання категорично не рекомендується. Якщо ви з будь-якої причини повторно використовуєте скрипти для попереднього випуску LFS, вам потрібно бути дуже обережними, щоб оновити скрипти відповідно до поточної версії книги LFS.

Ось декілька речей, які вам слід знати про створення кожного пакета:

- Деякі пакети патчуються перед компіляцією, але тільки тоді, коли патч необхідний для вирішення проблеми. Патч часто потрібен як у поточному, так і в наступних розділах, але іноді, коли той самий пакет компілюється більше ніж один раз, патч не потрібен відразу. Тому не хвилюйтеся, якщо інструкції для завантаженого патча здаються відсутніми. Під час застосування патча також можуть з'являтися

попереджувальні повідомлення про зміщення або нечіткість. Не хвилюйтесь про ці попередження; патч все одно був успішно застосований.

- Під час компіляції більшості пакетів на екрані з'являються деякі попередження. Це нормально і їх можна спокійно ігнорувати. Зазвичай ці попередження стосуються застарілого, але не недійсного використання синтаксису С або С++. Стандарти С досить часто змінюються, а деякі пакети ще не оновлені. Це не є серйозною проблемою, але призводить до появи попереджень.
- В останній раз перевірте, чи правильно налаштована змінна середовища `LFS`:

```
echo $LFS
```

Переконайтесь, що у вихідних даних відображається шлях до точки монтування розділу `LFS`, яка в нашому прикладі є `/mnt/lfs`.

- Наочанок слід підкреслити два важливі моменти:



Important

Інструкції з побудови передбачають, що вимоги до хост-системи, включаючи символічні посилання, були налагоджені належним чином:

- `bash` — це використовувана оболонка.
- `sh` — це символічне посилання на `bash`.
- `/usr/bin/awk` — це символічне посилання на `gawk`.
- `/usr/bin/yacc` — це символічне посилання на `bison` або на невеликий скрипт, який виконує `bison`.

Ось короткий опис процесу побудови.

1. Помістіть усі джерела та патчі в каталог, який буде доступний із середовища `chroot`, наприклад `/mnt/lfs/sources/`.
2. Перейдіть до каталогу `/mnt/lfs/sources/`.
3. Для кожного пакета:
 - a. За допомогою програми `tar` розпакуйте пакет, який потрібно зібрати. У розділах 5 і 6 переконайтесь, що ви є користувачем `/fs` під час розпакування пакета.
Не використовуйте жодних інших методів, крім команди `tar`, для розпакування вихідного коду. Зокрема, використання команди `cp -R` для копіювання дерева вихідного коду в інше місце може знищити часові мітки в дереві вихідного коду і привести до збою збірки.
 - b. Перейдіть до каталогу, створеного під час розпакування пакета.
 - c. Дотримуйтесь інструкцій щодо компіляції пакета.
 - d. Після завершення компіляції поверніться до каталогу `sources`.
 - e. Видаліть розпакований каталог пакету із `sources`, якщо не вказано інше.

Глава 5. Компіляція крос-інструментів

5.1. Вступ

У цьому розділі показано, як створити крос-компілятор та пов'язані з ним інструменти. Хоча тут крос-компіляція є імітованою, принципи її роботи такі самі, як і для справжнього крос-інструментарію.

Програми, скомпільовані в цьому розділі, будуть встановлені в каталозі `$LFS/tools`, щоб відокремити їх від файлів, встановлених у наступних розділах. Бібліотеки, з іншого боку, встановлюються в їх остаточне місце, оскільки вони стосуються системи, яку ми хочемо побудувати.

5.2. Binutils-2.44 - Етап 1

Пакет Binutils містить компілятор, асемблер та інші інструменти для обробки об'єктних файлів.

Приблизний час побудови: 1 SBU

Необхідний простір на диску: 677 MB

5.2.1. Встановлення крос-Binutils



Note

Поверніться і перечитайте примітки в розділі «Загальні інструкції щодо компіляції». Розуміння приміток, позначених як важливі, може вберегти вас від багатьох проблем у майбутньому.

Важливо, щоб Binutils був першим пакетом, що компілюється, оскільки як Glibc, так і GCC виконують різні тести на доступний лінкер та асемблер, щоб визначити, які з їхніх власних функцій слід увімкнути.

Документація Binutils рекомендує компілювати Binutils у спеціальному каталогі build:

```
mkdir -v build
cd      build
```



Note

Щоб значення SBU, наведені в іншій частині книги, були корисними, виміряйте час, необхідний для створення цього пакета від конфігурації до першої інсталяції включно. Щоб легко це зробити, оберніть команди в команду **time**, як показано нижче: **time { ./configure ... && make && make install; }**.

Тепер підготуйте Binutils до компіляції:

```
./configure --prefix=$LFS/tools \
            --with-sysroot=$LFS \
            --target=$LFS_TGT \
            --disable-nls \
            --enable-gprofng=no \
            --disable-werror \
            --enable-new-dtags \
            --enable-default-hash-style=gnu
```

Значення параметрів конфігурації:

--prefix=\$LFS/tools

Це вказує скрипту **configure** підготуватися до встановлення програм Binutils у каталозі **\$LFS/tools**.

--with-sysroot=\$LFS

Для крос-компіляції це вказує системі побудови шукати в **\$LFS** бібліотеки цільової системи за потреби.

--target=\$LFS_TGT

Оскільки опис машини в змінній **LFS_TGT** дещо відрізняється від значення, яке повертає скрипт **config.guess**, цей перемикач вказує скрипту **configure** налаштовувати систему збірки binutil для побудови крос-лінкера.

`--disable-nls`

Це вимикає інтернаціоналізацію, оскільки i18n не потрібна для тимчасових інструментів.

`--enable-gprofng=no`

Це вимикає побудову gprofng, який не потрібен для тимчасових інструментів.

`--disable-werror`

Це запобігає зупинці збірки у випадку, якщо компілятор хоста видає попередження.

`--enable-new-dtags`

Це змушує компонувач використовувати тег «runpath» для будовування шляхів пошуку бібліотек у виконувані файлі та спільні бібліотеки замість традиційного тегу «rpath». Це спрощує налагодження динамічно зв'язаних виконуваних файлів і дозволяє обійти потенційні проблеми в наборах тестів деяких пакетів.

`--enable-default-hash-style-gnu`

За замовчуванням, компонувач генерує як хеш-таблицю в стилі GNU, так і класичну хеш-таблицю ELF для спільніх бібліотек і динамічно зв'язаних виконуваних файлів. Хеш-таблиці призначені тільки для динамічного компонувача для виконання пошуку символів. У LFS динамічний компілятор (наданий пакетом Glibc) завжди використовує хеш-таблицю у стилі GNU, яка швидше виконує запити. Тому класична хеш-таблиця ELF є абсолютно непотрібною. Це змушує компілятор за замовчуванням генерувати тільки хеш-таблицю у стилі GNU, тому ми можемо уникнути втрати часу на генерацію класичної хеш-таблиці ELF під час побудови пакетів або втрати дискового простору для її зберігання.

Продовжте скомпілювавши пакет:

`make`

Встановіть пакет:

`make install`

Детальна інформація про цей пакет міститься в розділі 8.20.2, «Вміст Binutils».

5.3. GCC-14.2.0 - Етап 1

Пакет GCC містить колекцію компіляторів GNU, яка включає компілятори C і C++.

Приблизний час побудови: 3.2 SBU

Необхідний простір на диску: 4.8 GB

5.3.1. Інсталяція крос-GCC

GCC вимагає пакетів GMP, MPFR та MPC. Оскільки ці пакети можуть бути не включені до вашого хост-дистрибутива, вони будуть скомпільовані за допомогою GCC. Розпакуйте кожен пакет у каталогі GCC та перейменуйте отримані каталоги, щоб процедури компіляції GCC автоматично використовували їх:



Note

Щодо цього розділу часто виникають непорозуміння. Процедури такі самі, як і в інших розділах, як пояснено раніше (Інструкції з компіляції пакунків). Спочатку розпакуйте архів gcc-14.2.0 з каталогу sources, а потім перейдіть до створеного каталогу. Тільки після цього можна продовжувати виконувати наведені нижче інструкції.

```
tar -xf ../mpfr-4.2.1.tar.xz
mv -v mpfr-4.2.1 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.3.1.tar.gz
mv -v mpc-1.3.1 mpc
```

На хостах x86_64 встановіть ім'я каталогу за замовчуванням для 64-бітних бібліотек як «lib»:

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' \
      -i.orig gcc/config/i386/t-linux64
;;
esac
```

Документація GCC рекомендує компілювати GCC у окремому каталогі build:

```
mkdir -v build
cd      build
```

Підготуйте GCC до компіляції:

```
./configure \
--target=$LFS_TGT \
--prefix=$LFS/tools \
--with-glibc-version=2.41 \
--with-sysroot=$LFS \
--with-newlib \
--without-headers \
--enable-default-pie \
--enable-default-ssp \
--disable-nls \
--disable-shared \
--disable-multilib \
--disable-threads \
--disable-libatomic \
--disable-libgomp \
--disable-libquadmath \
--disable-libssp \
--disable-libvtv \
--disable-libstdc++ \
--enable-languages=c,c++
```

Значення параметрів конфігурації:

--with-glibc-version=2.41

Ця опція визначає версію Glibc, яка буде використовуватися на цільовій системі. Вона не має відношення до libc дистрибутива хоста, оскільки все, що компілюється на етапі 1 GCC, буде працювати в середовищі chroot, яке ізольоване від libc дистрибутива хоста.

--with-newlib

Оскільки робоча бібліотека С ще не доступна, це гарантує, що константа inhibit_libc буде визначена під час побудови libgcc. Це запобігає компіляції будь-якого коду, який вимагає підтримки libc.

--without-headers

Під час створення повного крос-компілятора GCC вимагає стандартних заголовків, сумісних із цільовою системою. Для наших цілей ці заголовки не будуть потрібні. Цей перемикач запобігає пошуку їх GCC.

--enable-default-pie та --enable-default-ssp

Ці перемикачі дозволяють GCC компілювати програми з деякими функціями посилення безпеки (більше інформації про них у примітці про PIE та SSP у розділі 8) за замовчуванням. На цьому етапі вони не є строго необхідними, оскільки компілятор буде створювати лише тимчасові виконувані файли. Але чистіше, щоб тимчасові пакети були якомога близчими до остаточних.

--disable-shared

Цей параметр змушує GCC статично зв'язувати свої внутрішні бібліотеки. Це потрібно, оскільки спільні бібліотеки вимагають Glibc, яка ще не встановлена на цільовій системі.

--disable-multilib

На x86_64 LFS не підтримує конфігурацію multilib. Цей перемикач нешкідливий для x86.

```
--disable-threads, --disable-libatomic, --disable-libgomp, --disable-libquadmath,
--disable-libssp, --disable-libvtv, --disable-libstdc++
```

Ці перемикачі вимикають підтримку багатопотоковості, libatomic, libgomp, libquadmath, libssp, libvtv та стандартної бібліотеки C++ відповідно. Ці функції можуть не компілюватися під час побудови крос-компілятора і не є необхідними для завдання крос-компіляції тимчасової libc.

```
--enable-languages=c,c++
```

Ця опція гарантує, що будуть побудовані тільки компілятори С і С++. Це єдині мови, які зараз потрібні.

Зкомпілюйте запустивши:

```
make
```

Встановіть пакет:

```
make install
```

Ця збірка GCC встановила кілька внутрішніх системних заголовків. Зазвичай один з них, limits.h, у свою чергу, включає відповідний системний заголовок limits.h, в даному випадку \$LFS/usr/include/limits.h. Однак на момент компіляції GCC \$LFS/usr/include/limits.h не існує, тому внутрішній заголовок, який щойно був встановлений, є частковим, самостійним файлом і не включає розширені функції системного заголовка. Це достатньо для побудови Glibc, але пізніше знадобиться повний внутрішній заголовок. Створіть повну версію внутрішнього заголовка за допомогою команди, яка ідентична тій, яку використовує система побудови GCC за звичайних обставин:



Note

У наведеному нижче прикладі вводу командного рядка показано вкладену заміну команди з використанням двох методів: зворотних лапок і конструкції \$(). Його можна було б переписати, використовуючи один і той самий метод для обох замін, але він показаний у такому вигляді, щоб продемонструвати, як їх можна поєднувати. Зазвичай перевагу надають методу \$().

```
cd ..
cat gcc/limitx.h gcc/glimits.h gcc/limits.h > \
`dirname $($LFS_TGT-gcc -print-libgcc-file-name)`/include/limits.h
```

Детальна інформація про цей пакет міститься в розділі 8.29.2, «Вміст GCC».

5.4. Linux-6.13.4 API Headers

Заголовки API Linux (у `linux-6.13.4.tar.xz`) надають API ядра для використання Glibc.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 1.6 GB

5.4.1. Інсталяція Linux-6.13.4 API Headers

Ядро Linux повинно надати інтерфейс прикладного програмування (API) для використання системної бібліотеки C (Glibc в LFS). Це робиться шляхом очищення різних файлів заголовків C, які постачаються в tar архіві вихідного коду ядра Linux. Переконайтесь, що в пакеті немає застарілих файлів:

```
make mrproper
```

Тепер витягніть з вихідного коду заголовки ядра, видимі для користувача. Рекомендований об'єкт `make <headers_install>` не можна використовувати, оскільки він вимагає `rsync`, який може бути недоступним. Заголовки спочатку розміщуються в `./usr`, а потім копіюються в необхідне місце.

```
make headers
find usr/include -type f ! -name '*.*' -delete
cp -rv usr/include $LFS/usr
```

5.4.2. Вміст Linux-6.13.4 API Headers

Встановлені заголовки: /usr/include/asm/*.h, /usr/include/asm-generic/*.h, /usr/include/drm/*.h, /usr/include/linux/*.h, /usr/include/misc/*.h, /usr/include/mtd/*.h, /usr/include/rdma/*.h, /usr/include/scsi/*.h, /usr/include/sound/*.h, /usr/include/video/*.h, and /usr/include/xen/*.h

Встановлені директорії: /usr/include/asm, /usr/include/asm-generic, /usr/include/drm, /usr/include/linux, /usr/include/misc, /usr/include/mtd, /usr/include/rdma, /usr/include/scsi, /usr/include/sound, /usr/include/video, and /usr/include/xen

Короткий опис

/usr/include/asm/*.h	The Linux API ASM Headers
/usr/include/asm-generic/*.h	The Linux API ASM Generic Headers
/usr/include/drm/*.h	The Linux API DRM Headers
/usr/include/linux/*.h	The Linux API Linux Headers
/usr/include/misc/*.h	The Linux API Miscellaneous Headers
/usr/include/mtd/*.h	The Linux API MTD Headers
/usr/include/rdma/*.h	The Linux API RDMA Headers
/usr/include/scsi/*.h	The Linux API SCSI Headers
/usr/include/sound/*.h	The Linux API Sound Headers
/usr/include/video/*.h	The Linux API Video Headers
/usr/include/xen/*.h	The Linux API Xen Headers

5.5. Glibc-2.41

Пакет Glibc містить основну бібліотеку С. Ця бібліотека надає основні процедури для виділення пам'яті, пошуку каталогів, відкриття та закриття файлів, читання та запису файлів, обробки рядків, зіставлення зразків, арифметичних операцій тощо.

Приблизний час побудови: 1.4 SBU

Необхідний простір на диску: 850 MB

5.5.1. Інсталяція Glibc

Спочатку створіть символічне посилання для відповідності LSB. Крім того, для x86_64 створіть символічне посилання сумісності, необхідне для належної роботи завантажувача динамічної бібліотеки:

```
case $ (uname -m) in
    i?86) ln -sfv ld-linux.so.2 $LFS/lib/ld-lsb.so.3
    ;;
    x86_64) ln -sfv ../lib/ld-linux-x86-64.so.2 $LFS/lib64
              ln -sfv ../lib/ld-linux-x86-64.so.2 $LFS/lib64/ld-lsb-x86-64.so.3
    ;;
esac
```

Note

Вищезазначена команда є правильною. Команда **In** має кілька синтаксичних варіантів, тому перед тим, як повідомляти про те, що може здаватися помилкою, обов'язково перевірте **infocoreutils In** та **In(1)**.

Деякі програми Glibc використовують несумісний з FHS каталог /var/db для зберігання даних під час виконання. Застосуйте наступний патч, щоб змусити такі програми зберігати свої дані виконання у місцях, сумісних з FHS:

```
patch -Np1 -i ./glibc-2.41-fhs-1.patch
```

У документації до Glibc рекомендується збирати Glibc у окремому каталозі build:

```
mkdir -v build
cd      build
```

Переконайтесь, що утиліти **ldconfig** та **sln** встановлено у /usr/sbin:

```
echo "rootsbindir=/usr/sbin" > configparms
```

Далі підготуйте Glibc до компіляції:

```
./configure
--prefix=/usr
--host=$LFS_TGT
--build=$../scripts/config.guess
--enable-kernel=5.4
--with-headers=$LFS/usr/include
--disable-nscd
libc_cv_slibdir=/usr/lib
```

Значення параметрів конфігурації:

`--host=$LFS_TGT, --build=$./scripts/config.guess)`

Комбінований ефект цих перемикачів полягає у тому, що система збірки Glibc налаштовується на крос-компіляцію, використовуючи крос-лінкер та крос-компілятор у `$LFS/tools`.

`--enable-kernel=5.4`

Це вказує Glibc скомпілювати бібліотеку з підтримкою ядер Linux версії 5.4 і пізніших. Обхідні шляхи для старіших ядер не вмикаються.

`--with-headers=$LFS/usr/include`

Це вказує Glibc компілювати себе з урахуванням заголовків, нещодавно встановлених до каталогу `$LFS/usr/include`, щоб точно знати, які особливості має ядро, і оптимізувати себе відповідно.

`libc_cv_slibdir=/usr/lib`

Це гарантує, що бібліотеку буде встановлено до каталогу `/usr/lib`, а не до каталогу за замовчуванням `/lib64` на 64-роздрядних машинах.

`--disable-nscd`

Не збирати демона кешу служби імен, який більше не використовується.

На цьому етапі може з'явитися наступне попередження:

```
configure: WARNING:
*** These auxiliary programs are missing or
*** incompatible versions: msgfmt
*** some features will be disabled.
*** Check the INSTALL file for required versions.
```

Відсутня або несумісна програма `msgfmt` зазвичай нешкідлива. Програма `msgfmt` є частиною пакунка Gettext, який має бути у складі дистрибутива.



Note

Було повідомлення, що цей пакунок може не працювати під час збирання у режимі «паралельного make». Якщо це сталося, повторно запустіть команду `make` з параметром `-j1`.

Зкомпілюйте пакет:

`make`

Встановіть пакет:



Warning

Якщо LFS налаштовано неправильно, і, незважаючи на рекомендації, ви збираєте від імені користувача `root`, наступна команда встановить щойно зібраний Glibc у вашу хост-систему, що майже напевно зробить його непридатним для використання. Тому перед виконанням наступної команди двічі перевірте, чи правильно налаштовано оточення і чи не є ви користувачем `root`.

`make DESTDIR=$LFS install`

Значення параметра make install:

```
DESTDIR=$LFS
```

Змінна `DESTDIR` використовується майже всіма пакунками для визначення місця, куди пакунок має бути встановлений. Якщо її не задано, за замовчуванням вона буде встановлена у кореневий каталог (/). Тут ми вкажемо, що пакунок буде встановлено до `$LFS`, який стане кореневим каталогом у Розділ 7.4, «Вхід до середовища Chroot».

Виправити жорстке кодування шляху до виконуваного завантажувача у скрипті `ldd`:

```
sed '/RTLDLIST=/s@/usr@@g' -i $LFS/usr/bin/ldd
```



Caution

На цьому етапі необхідно зупинитися і переконатися, що основні функції (компіляція і компонування) нового інструментарію працюють належним чином. Щоб виконати перевірку працевздатності, виконайте наступні команди:

```
echo 'int main() {}' | $LFS_TGT-gcc -xc -
readelf -l a.out | grep ld-linux
```

Якщо все працює коректно, помилок бути не повинно, а висновок останньої команди буде виглядати як:

```
[Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
```

Зауважте, що для 32-роздрядних машин ім'я інтерпретатора буде `/lib/ld-linux.so.2`.

Якщо вивід не такий, як показано вище, або взагалі відсутній, значить щось не так. Дослідіть і повторіть кроки, щоб з'ясувати, у чому проблема, і виправити її. Цю проблему потрібно вирішити, перш ніж продовжувати.

Коли все буде гаразд, очистіть файл тесту:

```
rm -v a.out
```



Note

Збирання пакунків у наступному розділі слугуватиме додатковою перевіркою того, що набір інструментів було зібрано належним чином. Якщо якийсь пакунок, особливо Binutils-pass2 або GCC-pass2, не вдається зібрати, це свідчить про те, що щось пішло не так під час попереднього встановлення Binutils, GCC або Glibc.

Докладні відомості про цей пакунок наведено у Розділі 8.5.3, «Вміст Glibc».

5.6. Libstdc++ із GCC-14.2.0

Libstdc++ - це стандартна бібліотека C++. Вона необхідна для компіляції C++ коду (частина GCC написана на C++), але ми відкладали встановлення, коли збиралі gcc-pass1, оскільки Libstdc++ залежить від Glibc, який ще не був доступний у цільовому каталозі

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 850 MB

5.6.1. Встановлення цільової бібліотеки Libstdc++



Note

Libstdc++ є частиною вихідного коду GCC. Спочатку слід розпакувати tar-архів GCC і перейти до каталогу gcc-14.2.0.

Створіть окремий каталог збірки для Libstdc++ і увійдіть до нього:

```
mkdir -v build
cd      build
```

Підготуйте Libstdc++ до компіляції:

```
./libstdc++-v3/configure \
--host=$LFS_TGT \
--build=$(./config.guess) \
--prefix=/usr \
--disable-multilib \
--disable-nls \
--disable-libstdcxx-pch \
--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/14.2.0
```

Значення параметрів конфігурації:

--host=...

Вказує, що крос-компілятор, який ми щойно зібрали, слід використовувати замість того, що знаходиться у /usr/bin.

--disable-libstdcxx-pch

Цей ключ запобігає встановленню попередньо скомпільованих файлів include, які не потрібні на даному етапі.

--with-gxx-include-dir=/tools/\$LFS_TGT/include/c++/14.2.0

Вказує каталог встановлення для файлів include. Оскільки Libstdc++ є стандартною бібліотекою C++ для LFS, цей каталог має відповідати місцю, де компілятор C++ (**\$LFS_TGT-g++**) шукатиме стандартні C++ файли include. У звичайній збірці ця інформація автоматично передається до **configure** параметрів Libstdc++ з каталогу верхнього рівня. У нашому випадку цю інформацію слід вказати явно. Компілятор C++ додасть шлях до кореневого каталогу \$LFS (вказаній під час збирання GCC-pass1) до шляху пошуку файлів включення, тому він фактично шукати у \$LFS/tools/\$LFS_TGT/include/c++/14.2.0. Поєднання змінної DESTDIR (у команді **make install** нижче) і цього перемикача призведе до встановлення заголовків саме туди.

Зкомпілюйте Libstdc++:

```
make
```

Встановіть бібліотеку:

```
make DESTDIR=$LFS install
```

Видаліть файли архіву libtool, оскільки вони заважають крос-компіляції:

```
rm -v $LFS/usr/lib/lib{stdc++,exp,fs},supc++.la
```

Детальна інформація про цей пакет міститься у Розділі 8.29.2, «Зміст GCC».

Глава 6. Крос-компіляція тимчасових інструментів

6.1. Вступ

У цьому розділі показано, як крос-компілювати основні утиліти за допомогою щойно створеного крос-ланцюжка інструментів. Ці утиліти встановлено у їх остаточне розташування, але їх ще не можна використовувати. Основні завдання все ще покладаються на інструменти хоста. Тим не менш, встановлені бібліотеки використовуються при компонуванні.

Використання утиліт стане можливим у наступній главі після входу у середовище «`chroot`». Але всі пакунки, зібрані у цій главі, мають бути зібрані до того, як ми це зробимо. Тому поки що ми не можемо бути незалежними від хост-системи.

Ще раз нагадаємо, що неправильне налаштування `lfs` разом зі збиранням від імені користувача `root` може вивести ваш комп'ютер з ладу. Весь розділ має бути виконано від імені користувача `lfs`, у середовищі, описаному у Розділі 4.4, «Налаштування оточення».

6.2. M4-1.4.19

Пакет M4 містить макропроцесор.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 32 MB

6.2.1. Інсталяція M4

Підготуйте M4 до компіляції:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Детальна інформація про цей пакет знаходиться в Розділі 8.13.2, «Вміст M4».

6.3. Ncurses-6.5

Пакет Ncurses містить бібліотеки для управління текстовими екранами, незалежного від типу терміналу.

Приблизний час побудови: 0.4 SBU

Необхідний простір на диску: 53 MB

6.3.1. Інсталяція Ncurses

Спочатку виконайте наступні команди, щоб зібрати програму «tic» на хості збірки:

```
mkdir build
pushd build
    ./configure AWK=gawk
    make -C include
    make -C progs tic
popd
```

Підготуйте Ncurses до компіляції:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(./config.guess) \
            --mandir=/usr/share/man \
            --with-manpage-format=normal \
            --with-shared \
            --without-normal \
            --with-cxx-shared \
            --without-debug \
            --without-ada \
            --disable-stripping \
            AWK=gawk
```

Значення параметрів конфігурації:

--with-manpage-format=normal

Це не дозволяє Ncurses встановлювати стиснуті сторінки посібника, що може статися, якщо сам дистрибутив має стиснуті сторінки посібника.

--with-shared

Дозволяє Ncurses збирати і встановлювати спільні бібліотеки C.

--without-normal

Дозволяє Ncurses збирати і встановлювати статичні бібліотеки C.

--without-debug

Дозволяє Ncurses збирати і встановлювати налагоджувальні бібліотеки.

--with-cxx-shared

Дозволяє Ncurses збирати і встановлювати спільні прив'язки C++. Також забороняє збирати і встановлювати статичні зв'язки C++ прив'язки.

--without-ada

Це гарантує, що Ncurses не збиратиме підтримку компілятора Ada, який може бути присутнім на хості, але буде недоступним, коли ми увійдемо до середовища **chroot**.

--disable-stripping

Цей перемикач забороняє системі збирати використовувати програму **strip** з хоста. Використання інструментів хоста у крос-скомпільованих програмах, може призвести до збою.

AWK=gawk

Цей перемикач забороняє системі збирати використовувати програму **mawk** з хоста. Деякі версії **mawk** можуть призвести до того, що цей пакунок не вдасться зібрати.

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS TIC_PATH=$(pwd)/build/progs/tic install
ln -sv libncursesw.so $LFS/usr/lib/libncurses.so
sed -e 's/^#if.*XOPEN.*$/#if 1/' \
-i $LFS/usr/include/curses.h
```

Значення параметрів інсталяції:

TIC_PATH=\$(pwd)/build/progs/tic

Нам потрібно передати шлях до новозбудованої програми **tic**, яка працює на машині збірки, щоб база даних терміналу могла бути створена без помилок.

In -sv libncursesw.so \$LFS/usr/lib/libncurses.so

Бібліотека **libncurses.so** потрібна кільком пакункам, які ми незабаром зберемо. Ми створимо це посилання, щоб використовувати **libncursesw.so** як заміну.

sed -e 's/^#if.*XOPEN.*\$/#if 1/' ...

Заголовний файл **curses.h** містить визначення різних структур даних Ncurses. З різними макроозначеннями препроцесора можна використовувати два різні набори визначень структур даних: 8-бітне визначення сумісне з **libncurses.so**, а широко-символьне визначення сумісне з **libncursesw.so**. Оскільки ми використовуємо **libncursesw.so** замість **libncurses.so**, відредагуйте заголовний файл так, щоб він завжди використовував широко-символьне визначення структури даних, сумісне з **libncursesw.so**.

Докладні відомості про цей пакунок наведено у Розділі 8.30.2, «Зміст Ncurses».

6.4. Bash-5.2.37

Пакет Bash містить Bourne-Again Shell.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 68 MB

6.4.1. Інсталяція Bash

Підготуйте Bash до компіляції:

```
./configure --prefix=/usr
            --build=$(sh support/config.guess)
            --host=$LFS_TGT
            --without-bash-malloc
```

Значення параметрів конфігурації:

--without-bash-malloc

Цей параметр вимикає використання функції виділення пам'яті Bash (`malloc`), яка, як відомо, спричиняє помилки сегментації. Якщо вимкнути цей параметр, Bash використовуватиме функції `malloc` з Glibc, які є більш стабільними.

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Зробіть посилання для програм, які використовують `sh` для оболонки:

```
ln -sv bash $LFS/bin/sh
```

Докладні відомості про цей пакунок наведено у Розділі 8.36.2, «Вміст Bash».

6.5. Coreutils-9.6

Пакет Coreutils містить основні утиліти, необхідні для кожної операційної системи.

Приблизний час побудови: 0.3 SBU

Необхідний простір на диску: 181 MB

6.5.1. Інсталяція Coreutils

Підготуйте Coreutils до компіляції:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess) \
            --enable-install-program=hostname \
            --enable-no-install-program=kill,uptime
```

Значення параметрів конфігурації:

--enable-install-program=hostname

Це дозволить зібрати і встановити двійковий файл **hostname** - за замовчуванням його вимкнено, але він потрібен для тестового набору Perl.

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Перемістіть програми до їх остаточного очікуваного розташування. Хоча у цьому тимчасовому середовищі це не обов'язково, ми повинні це зробити, оскільки деякі програми жорстко кодують розташування виконуваних файлів:

```
mv -v $LFS/usr/bin/chroot           $LFS/usr/sbin
mkdir -pv $LFS/usr/share/man/man8
mv -v $LFS/usr/share/man/man1/chroot.1   $LFS/usr/share/man/man8/chroot.8
sed -i 's/"1"/"8"/'                  $LFS/usr/share/man/man8/chroot.8
```

Детальна інформація про цей пакет міститься у Розділі 8.58.2, «Вміст Coreutils».

6.6. Diffutils-3.11

Пакет Diffutils містить програми, які показують відмінності між файлами або каталогами.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 35 MB

6.6.1. Інсталяція Diffutils

Підготуйте Diffutils до компіляції:

```
./configure --prefix=/usr \
--host=$LFS_TGT \
--build=$(./build-aux/config.guess)
```

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Детальна інформація про цей пакет міститься в розділі 8.60.2, «Вміст Diffutils»

6.7. File-5.46

Пакет File містить утиліту для визначення типу заданого файлу або файлів.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 42 MB

6.7.1. Інсталяція File

Команда **file** на хості збірки має бути тієї ж версії, що і та, яку ми збираємо, щоб створити файл підпису.

Виконайте наступні команди, щоб створити тимчасову копію команди **file**:

```
mkdir build
pushd build
./configure --disable-bzlib \
--disable-libseccomp \
--disable-xzlib \
--disable-zlib
make
popd
```

Значення нових параметрів конфігурації:

--disable-*

Сценарій конфігурації намагається використати деякі пакунки з хост-дистрибутива, якщо відповідні файли бібліотек існують. Це може привести до помилки компіляції, якщо файл бібліотеки існує, а відповідних заголовкових файлів немає. Ці опції запобігають використанню цих непотрібних можливостей хоста.

Підготуйте File до компіляції:

```
./configure --prefix=/usr --host=$LFS_TGT --build=$ ./config.guess
```

Зкомпілюйте пакет:

```
make FILE_COMPILE=$ (pwd) /build/src/file
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Видаліть файл архіву libtool, оскільки він заважає крос-компіляції:

```
rm -v $LFS/usr/lib/libmagic.la
```

Детальну інформацію про цей пакунок наведено у Розділі 8.11.2, «Вміст File».

6.8. Findutils-4.10.0

Пакет Findutils містить програми для пошуку файлів. Програми надаються для пошуку всіх файлів у дереві каталогу, а також для створення, підтримки та пошуку в базі даних (часто швидше, ніж рекурсивний пошук, але ненадійно, якщо база даних не оновлювалася останнім часом). Findutils також постачає програму **xargs**, за допомогою якої можна виконати вказану команду для кожного файлу, знайденого під час пошуку.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 48 MB

6.8.1. Інсталяція Findutils

Підготуйте Findutils до компіляції:

```
./configure --prefix=/usr \
            --localstatedir=/var/lib/locate \
            --host=$LFS_TGT \
            --build=$ (build-aux/config.guess)
```

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Детальна інформація про цю упаковку міститься в Розділі 8.62.2, «Вміст Findutils».

6.9. Gawk-5.3.1

Пакет Gawk містить програми для роботи з текстовими файлами.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 47 MB

6.9.1. Інсталяція Gawk

По-перше, переконайтесь, що деякі непотрібні файли не інстальовано:

```
sed -i 's/extras//' Makefile.in
```

Підготуйте Gawk до компіляції:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Детальніше про цей пакунок наведено у Розділі 8.61.2, «Вміст Gawk».

6.10. Grep-3.11

Пакет Grep містить програми для пошуку у вмісті файлів.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 27 MB

6.10.1. Інсталляція Grep

Підготуйте Grep до компіляції:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(./build-aux/config.guess)
```

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Детальніше про цей пакунок наведено у Розділі 8.35.2, «Вміст Grep».

6.11. Gzip-1.13

Пакет Gzip містить програми для стиснення та розпакування файлів.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 11 MB

6.11.1. Інсталляція Gzip

Підготуйте Gzip до компіляції:

```
./configure --prefix=/usr --host=$LFS_TGT
```

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Детальніше про цей пакунок наведено у Розділі 8.65.2, «Вміст Gzip».

6.12. Make-4.4.1

До складу пакунка Make входить програма для керування генерацією виконуваних файлів та інших невихідних файлів пакунка з вихідних файлів.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 15 MB

6.12.1. Інсталляція Make

Підготуйте Grep до компіляції:

```
./configure --prefix=/usr \
            --without-guile \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Значення нових параметрів конфігурації:

--without-guile

Хоча ми виконуємо крос-компіляцію, configure намагається використати guile з хосту збірки, якщо знаходить його. Це призводить до невдалої компіляції, тому цей перемикач забороняє його використання.

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Детальніше про цей пакунок наведено у Розділі 8.69.2, «Вміст Make».

6.13. Patch-2.7.6

Пакунок Patch містить програму для модифікації або створення файлів шляхом застосування файлу «patch», який зазвичай створюється за допомогою програми **diff**.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 12 MB

6.13.1. Інсталяція Patch

Підготуйте Patch до компіляції:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$ (build-aux/config.guess)
```

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Детальніше про цей пакунок наведено у Розділі 8.70.2, «Вміст Patch».

6.14. Sed-4.9

Пакет Sed містить редактор потоків.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 21 MB

6.14.1. Інсталляція Sed

Підготуйте Sed до компіляції:

```
./configure --prefix=/usr \
--host=$LFS_TGT \
--build=$(./build-aux/config.guess)
```

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Детальніше про цей пакунок наведено у Розділі 8.31.2, «Вміст Sed».

6.15. Tar-1.35

Пакет Tar надає можливість створювати tar-архіви, а також виконувати різні інші види маніпуляцій з архівами. Tar можна використовувати на раніше створених архівах для вилучення файлів, зберігання додаткових файлів, а також для оновлення або переліку файлів, які вже було збережено.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 42 MB

6.15.1. Інсталляція Tar

Підготуйте Tar до компіляції:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$ (build-aux/config.guess)
```

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Детальніше про цей пакунок наведено у Розділі 8.71.2, «Вміст Tar».

6.16. Xz-5.6.4

Пакет Xz містить програми для стиснення та розпакування файлів. Він надає можливості для lzma та новіших форматів стиснення xz. Стиснення текстових файлів за допомогою xz дає кращий відсоток стиснення, ніж традиційними командами **gzip** або **bzip2**.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 21 MB

6.16.1. Інсталляція Xz

Підготуйте Xz до компіляції:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=${build-aux/config.guess} \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.6.4
```

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Детальніше про цей пакунок наведено у Розділі 8.8.2, «Вміст Xz».

6.17. Binutils-2.44 - Pass 2

Пакет Binutils містить компонувальник, асемблер та інші інструменти для роботи з об'єктними файлами.

Приблизний час побудови: 0.4 SBU

Необхідний простір на диску: 539 MB

6.17.1. Інсталяція Binutils

Система збірки Binutils покладається на копію libtool, що постачається, для компонування з внутрішніми статичними бібліотеками, але копії libiberty та zlib, що постачаються у пакунку, не використовують libtool. Ця невідповідність може привести до того, що створені двійкові файли буде помилково з'єднано з бібліотеками з основного дистрибутива. Виправлення цієї проблеми:

```
sed '6031s/$add_dir//' -i ltmain.sh
```

Знову створіть окремий каталог build:

```
mkdir -v build
cd      build
```

Підготуйте Binutils до компіляції:

```
./configure \
  --prefix=/usr \
  --build=$(.. /config.guess) \
  --host=$LFS_TGT \
  --disable-nls \
  --enable-shared \
  --enable-gprofng=no \
  --disable-werror \
  --enable-64-bit-bfd \
  --enable-new-dtags \
  --enable-default-hash-style=gnu
```

Значення нових параметрів конфігурації:

--enable-shared

Збирає libbfd як спільну бібліотеку.

--enable-64-bit-bfd

Вмикає 64-бітну підтримку (на хостах з меншим розміром слова). Це може бути непотрібним на 64-бітних системах, але не завдасть жодної шкоди.

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make DESTDIR=$LFS install
```

Видаліть архівні файли libtool, оскільки вони заважають крос-компіляції, і видаліть непотрібні статичні бібліотеки:

```
rm -v $LFS/usr/lib/lib{bfd,ctf,ctf-nobfd,opcodes,sframe}.{a,la}
```

Детальніше про цей пакунок наведено у Розділі 8.20.2, «Вміст Binutils».

6.18. GCC-14.2.0 - Pass 2

Пакунок GCC містить колекцію компіляторів GNU, до якої входять компілятори C та C++.

Приблизний час побудови: 4.1 SBU

Необхідний простір на диску: 5.5 GB

6.18.1. Інсталляція GCC

Як і в першій збірці GCC, потрібні пакунки GMP, MPFR та MPC. Розпакуйте tar-архіви і перемістіть їх до потрібних каталогів:

```
tar -xf ../mpfr-4.2.1.tar.xz
mv -v mpfr-4.2.1 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.3.1.tar.gz
mv -v mpc-1.3.1 mpc
```

Якщо ви збираєте на x86_64, змініть назву каталогу для 64-бітних бібліотек за замовчуванням на «lib»:

```
case $(uname -m) in
x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
;;
esac
```

Перевизначити правило збирання заголовків libgcc та libstdc++, щоб дозволити збирання цих бібліотек з підтримкою потоків POSIX:

```
sed '/thread_header =/s/@.*@/gthr-posix.h/' \
    -i libgcc/Makefile.in libstdc++-v3/include/Makefile.in
```

Знову створіть окремий каталог build:

```
mkdir -v build
cd      build
```

Підготуйте GCC до компіляції:

```
./configure
--build=$(.. /config.guess)
--host=$LFS_TGT
--target=$LFS_TGT
LDFLAGS_FOR_TARGET=-L$PWD/$LFS_TGT/libgcc
--prefix=/usr
--with-build-sysroot=$LFS
--enable-default-pie
--enable-default-ssp
--disable-nls
--disable-multilib
--disable-libatomic
--disable-libgomp
--disable-libquadmath
--disable-lbsanitizer
--disable-libssp
--disable-libvtv
--enable-languages=c, c++
```

Значення нових параметрів конфігурації:*--with-build-sysroot=\$LFS*

Зазвичай, використання *--host* гарантує, що для збирання GCC використовується крос-компілятор, і що компілятор знає, що він має шукати заголовки та бібліотеки у *\$LFS*. Але система збирання GCC використовує інші інструменти, які не знають про це місце. Цей перемикач потрібен для того, щоб ці інструменти знаходили потрібні файли у *\$LFS*, а не на хості.

--target=\$LFS_TGT

Ми виконуємо крос-компіляцію GCC, тому неможливо зібрати цільові бібліотеки (*libgcc* і *libstdc++*) з бінарними файлами GCC, скомпільованими за допомогою цієї передачі - ці бінарні файли не будуть виконуватися на хості. За замовчуванням система збирання GCC намагатиметься використати компілятори С і С++ на хості як обхідний шлях. Збірка цільових бібліотек GCC з іншою версією GCC не підтримується, тому використання компіляторів хоста може привести до невдалого завершення збірки. Цей параметр гарантує, що бібліотеки буде зібрано за допомогою GCC етапу 1.

LDFLAGS_FOR_TARGET=...

Дозволяє *libstdc++* використовувати *libgcc*, що збирається за цим етапом, замість попередньої версії, що збирається за *gcc-pass1*. Попередня версія не може належним чином підтримувати обробку винятків у С++, оскільки її було зібрано без підтримки *libc*.

--disable-lbsanitizer

Вимкнути бібліотеки часу виконання GCC-санітайзерів. Вони не потрібні для тимчасового встановлення. У *gcc-pass1* це було, що малося на увазі під *--disable-libstdcxx*, а тепер ми можемо передати його явно.

Зкомпілюйте пакет:

make

Встановіть пакет:

make DESTDIR=\$LFS install

На завершення створіть символічне посилання на утиліту. Багато програм і скриптів виконують `cc` замість `gcc`, який використовується для збереження універсальності програм, а отже, їх можна використовувати на всіх типах UNIX-систем, де не завжди встановлено компілятор C GNU. Запуск `cc` залишає системному адміністратору свободу вибору компілятора C для встановлення:

```
ln -sv gcc $LFS/usr/bin/cc
```

Детальніше про цей пакунок наведено у Розділі 8.29.2, «Вміст GCC».

Глава 7. Вступ до Chroot та створення додаткових тимчасових інструментів

7.1. Вступ

У цьому розділі показано, як зібрати останні відсутні біти тимчасової системи: інструменти, необхідні для складання різних пакунків. Тепер, коли всі циклічні залежності вирішено, для збірки можна використовувати середовище «chroot», повністю ізольоване від операційної системи хоста (за винятком працюючого ядра).

Для правильної роботи ізольованого середовища необхідно встановити певний зв'язок з працюючим ядром. Це робиться за допомогою так званих *Virtual Kernel File Systems*, які буде змонтовано перед входом до середовища chroot. Ви можете перевірити, чи їх змонтовано, виконавши команду **findmnt**.

До Розділу 7.4, «Входження до середовища chroot», команди слід виконувати від імені користувача `root` зі встановленою змінною `LFS`. Після входу у chroot усі команди будуть виконуватися від імені користувача `root`, на щастя, без доступу до операційної системи комп'ютера, на якому ви збирали LFS. У будь-якому випадку, будьте обережні, оскільки можна легко зруйнувати всю систему LFS поганими командами.

7.2. Зміна власника



Note

Команди, наведені у цій книзі, слід виконувати, увійшовши до системи як користувач `root`, а не як користувач `lfs`. Крім того, перевірте, чи встановлено параметр `$LFS` у середовищі користувача `root`.

Наразі вся ієрархія каталогів у `$LFS` належить користувачеві `lfs`, користувачеві, який існує лише на хост-системі. Якщо каталоги і файли у `$LFS` залишили без змін, вони належатимуть користувачеві з ідентифікатором користувача без відповідного облікового запису. Це небезпечно, оскільки обліковий запис користувача, створений пізніше, може отримати той самий ідентифікатор користувача і володіти усіма файлами у `$LFS`, що зробить ці файли вразливими до можливих зловмисних маніпуляцій.

Щоб вирішити цю проблему, змініть права доступу до каталогів `$LFS/*` на права користувача `root`, виконавши наступну команду:

```
chown --from lfs -R root:$root $LFS/{usr,lib,var,etc,bin,sbin,tools}
case $(uname -m) in
  x86_64) chown --from lfs -R root:$root $LFS/lib64 ;;
esac
```

7.3. Підготовка Virtual Kernel File Systems

Програми, що працюють у користувальницькому просторі, використовують різні файлові системи, створені ядром для взаємодії з самим ядром. Ці файлові системи є віртуальними: для них не використовується місце на диску. Вміст цих файлових систем знаходиться у пам'яті. Ці файлові системи потрібно змонтувати у дереві каталогів `$LFS`, щоб програми могли знайти їх у середовищі chroot.

Почніть зі створення каталогів, до яких буде змонтовано ці віртуальні файлові системи:

```
mkdir -pv $LFS/{dev,proc,sys,run}
```

7.3.1. Монтаж та наповнення /dev

Під час звичайного завантаження системи LFS ядро автоматично монтує файлову систему `devtmpfs` у каталог `/dev`; ядро створює вузли пристроїв у цій віртуальній файловій системі під час процесу завантаження або при першому виявленні пристрою чи зверненні до нього. Демон `udev` може змінювати права власності або дозволи для вузлів пристроїв, створених ядром і створювати нові вузли пристроїв або символічні посилання, щоб полегшити роботу мейнтайнерів дистрибутива та системних адміністраторів. (Докладно див. Розділ 9.3.2.2, «Створення вузлів пристроїв».) Якщо ядро хоста підтримує `devtmpfs`, ми можемо просто змонтувати `devtmpfs` у `$LFS/dev` і покластися на ядро для його заповнення.

Але у деяких ядрах хостів відсутня підтримка `devtmpfs`; ці хост-дистрибутиви використовують різні методи для створення вмісту каталогу `/dev`. Отже, єдиним способом заповнення каталогу `$LFS/dev`, який не залежить від хоста, є прив'язування каталогу `/dev` хост-системи. Прив'язування - це спеціальний тип монтування, який робить піддерево каталогу або файл видимим у іншому місці. Для цього скористайтеся командою.

```
mount -v --bind /dev $LFS/dev
```

7.3.2. Монтування Virtual Kernel File Systems

Тепер змонтуйте решту файлових систем віртуального ядра:

```
mount -vt devpts devpts -o gid=5,mode=0620 $LFS/dev/pts
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
mount -vt tmpfs tmpfs $LFS/run
```

Значення параметрів монтування для devpts:

gid=5

Це гарантує, що усім створеним `devpts` вузлам пристроїв буде належати ідентифікатор групи 5. Цей ідентифікатор ми будемо використовувати пізніше для групи `tty`. Ми використовуємо ідентифікатор групи замість імені, оскільки хост-система може використовувати інший ідентифікатор для групи `tty`.

mode=0620

Це гарантує, що всі вузли пристроїв, створені за допомогою `devpts`, матимуть режим 0620 (для читання і запису користувачем, для запису групою). Разом з наведеним вище параметром це гарантує, що `devpts` створюватиме вузли пристроїв, які відповідають вимогам `grantpt()`, тобто допоміжний двійковий файл Glibc `pt_chown` (який не встановлено за замовчуванням) не є необхідним.

У деяких хост-системах `/dev/shm` є символічним посиланням на каталог, зазвичай `/run/shm`. `Tmpfs` з каталогу `/run` було змонтовано над, тому у цьому випадку потрібно створити лише каталог з відповідними правами доступу.

На інших хост-системах `/dev/shm` є точкою монтування для `tmpfs`. У цьому випадку монтування `/dev` вище створить лише каталог `/dev/shm` як каталог у середовищі `chroot`. У цій ситуації ми повинні явно змонтувати `tmpfs`:

```
if [ -h $LFS/dev/shm ]; then
    install -v -d -m 1777 $LFS$(realpath /dev/shm)
else
    mount -vt tmpfs -o nosuid,nodev tmpfs $LFS/dev/shm
fi
```

7.4. Вхід до середовища Chroot

Тепер, коли всі пакунки, необхідні для збірки решти необхідних інструментів, встановлено у системі, настав час увійти до chroot-оточення і завершити встановлення тимчасових інструментів. Це середовище також буде використано для встановлення остаточної версії системи. Від імені користувача `root` виконайте наступну команду, щоб увійти до середовища, яке наразі містить лише тимчасові інструменти:

```
chroot "$LFS" /usr/bin/env -i \
    HOME=/root \
    TERM="$TERM" \
    PS1='(lfs chroot) \u:\w\$ ' \
    PATH=/usr/bin:/usr/sbin \
    MAKEFLAGS="-j$(nproc)" \
    TESTSUITEFLAGS="-j$(nproc)" \
    /bin/bash --login
```

Якщо ви не хочете використовувати усі доступні логічні ядра, замініть `$ (прогос)` на кількість логічних ядер, яку ви хочете використовувати для збирання пакунків у цій та наступних главах. На набори тестів деяких пакунків (зокрема, Autoconf, Libtool і Tar) у розділі 8 не впливають параметри `MAKEFLAGS`, натомість вони використовують змінну середовища `TESTSUITEFLAGS`. Ми встановлюємо її і тут для запуску цих тестових наборів на декількох ядрах.

Параметр `-i`, доданий до команди `env`, очистить усі змінні у середовищі chroot. Після цього знову будуть встановлені лише змінні `HOME`, `TERM`, `PS1` і `PATH`. Конструкція `TERM=$TERM` встановлює змінну `TERM` всередині chroot у те саме значення, що і поза chroot. Ця змінна необхідна для того, щоб програми на кшталт `vim` і `less` могли працювати належним чином. Якщо вам потрібні інші змінні, такі як `CFLAGS` або `CXXFLAGS`, це гарне місце для їх встановлення.

З цього моменту більше не потрібно використовувати змінну `LFS`, оскільки вся робота буде обмежена файлом `LFS` системи; команда `chroot` запускає оболонку Bash з кореневим (`/`) каталогом, встановленим в `$LFS`.

Зверніть увагу, що `/tools/bin` не знаходиться у `PATH`. Це означає, що набір крос-інструментів більше не використовуватиметься.

Також зверніть увагу, що у відповідь на запит `bash` буде сказано: `I have no name!` Це нормально, оскільки файл `/etc/passwd` ще не створено.

Note

Важливо, щоб усі команди у цій главі та наступних главах виконувалися у середовищі chroot. Якщо ви залишите це середовище з будь-якої причини (наприклад, перезавантаження), переконайтесь, що файлові системи віртуального ядра змонтовано, як описано у Розділі 7.3.1, «Монтування та заповнення `/dev`» та Розділі 7.3.2, «Монтування файлових систем віртуального ядра», і знову увійдіть до chroot перед продовженням встановлення.

7.5. Створення директорій

Настав час створити повну структуру каталогів у файловій системі LFS.



Note
Деякі з каталогів, згаданих у цьому розділі, можливо, вже було створено раніше за допомогою явних інструкцій або під час встановлення деяких пакунків. Для повноти інформації вони повторюються нижче.

Створіть кілька каталогів кореневого рівня, які не входять до обмеженого набору, описаного у попередніх розділах, виконавши наступну команду:

```
mkdir -pv {boot,home,mnt,opt,srv}
```

Створіть необхідний набір підкаталогів нижче кореневого рівня, виконавши наступні команди:

```
mkdir -pv /etc/{opt,sysconfig}
mkdir -pv /lib/firmware
mkdir -pv /media/{floppy,cdrom}
mkdir -pv /usr/{,local/}{include,src}
mkdir -pv /usr/lib/locale
mkdir -pv /usr/local/{bin,lib,sbin}
mkdir -pv /usr/{,local/}share/{color,dict,doc,info,locale,man}
mkdir -pv /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
mkdir -pv /var/{cache,local,log,mail,opt,spool}
mkdir -pv /var/lib/{color,misc,locate}

ln -sfv /run /var/run
ln -sfv /run/lock /var/lock

install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
```

За замовчуванням каталоги створюються з режимом доступу 755, але це не скрізь бажано. У наведених вище командах вносяться дві зміни - одна до домашнього каталогу користувача `root`, а інша - до каталогів для тимчасових файлів.

Перша зміна режиму гарантує, що не будь-хто може увійти до каталогу `/root` - так само, як звичайний користувач зробив би це у своєму домашньому каталозі. Друга зміна режиму гарантує, що будь-який користувач може записувати до каталогів `/tmp` і `/var/tmp`, але не може видаляти з них файли інших користувачів. Останнє заборонено так званим «липким бітом», старшим бітом (1) у 1777-бітній масці.

7.5.1. Зauważення щодо відповідності FHS

Це дерево каталогів базується на Стандарті ієархії файлової системи (FHS) (доступний за адресою <https://refspecs.linuxfoundation.org/fhs.shtml>). FHS також визначає необов'язковість існування додаткових каталогів, таких як `/usr/local/games` та `/usr/share/games`. У LFS ми створюємо лише ті каталоги, які дійсно необхідні. Втім, ви можете створити більше каталогів, якщо бажаєте.


Warning

FHS не вимагає існування каталогу /usr/lib64, і редактори LFS вирішили його не використовувати. Щоб інструкції в LFS і BLFS працювали правильно, необхідно, щоб цей каталог не існував. Час від часу вам слід перевіряти, що його не існує, тому що його легко створити випадково, і це, ймовірно, зламає вашу систему.

7.6. Створення необхідних файлів і символічних посилань

Історично склалося так, що Linux зберігає список змонтованих файлових систем у файлі /etc/mtab. Сучасні ядра підтримують цей список внутрішньо і надають його користувачеві у файловій системі /proc. Щоб задовольнити потреби утиліт, які очікують знайти /etc/mtab, створіть наступне символічне посилання:

```
ln -sv /proc/self/mounts /etc/mtab
```

Створіть базовий файл /etc/hosts, на який будуть посыпатися деякі тестові набори, а також один з конфігураційних файлів Perl:

```
cat > /etc/hosts << EOF
127.0.0.1 localhost $(hostname)
::1      localhost
EOF
```

Для того, щоб користувач root міг увійти в систему і щоб ім'я «root» розпізнавалося, повинні бути відповідні записи в файлах /etc/passwd і /etc/group.

Створіть файл /etc/passwd за допомогою наступної команди:

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/usr/bin/false
daemon:x:6:6:Daemon User:/dev/null:/usr/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/run/dbus:/usr/bin/false
uuidd:x:80:80:UUID Generation Daemon User:/dev/null:/usr/bin/false
nobody:x:65534:65534:Unprivileged User:/dev/null:/usr/bin/false
EOF
```

Актуальний пароль для користувача root буде встановлено пізніше.

Створіть файл /etc/group, виконавши наступну команду:

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:daemon
sys:x:2:
kmem:x:3:
tape:x:4:
tty:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
input:x:24:
mail:x:34:
kvm:x:61:
uuidd:x:80:
wheel:x:97:
users:x:999:
nogroup:x:65534:
EOF
```

Створені групи не є частиною жодного стандарту - це групи, визначені частково вимогами конфігурації Udev, описаної у розділі 9, а частково загальними домовленостями, що застосовуються у низці наявних дистрибутивів Linux. Крім того, деякі набори тестів покладаються на певних користувачів або групи. Стандартна база Linux (LSB, доступна за адресою <https://refspecs.linuxfoundation.org/lsb.shtml>) лише рекомендує, щоб, окрім root групи з ідентифікатором групи (GID), рівним 0, була присутня група bin з GID, рівним 1. GID 5 широко використовується для групи tty, і число 5 також використовується у /etc/fstab для файлової системи devpts. Усі інші назви груп та GID можуть бути вільно обрані системним адміністратором, оскільки добре написані програми не залежать від номерів GID, а використовують назву групи.

ID 65534 використовується ядром для NFS та окремих просторів імен користувачів для незіставлених користувачів і груп (які існують на сервері NFS або у батьківському просторі імен користувачів, але «не існують» на локальній машині або в окремому просторі імен). Ми присвоюємо nobody і nogroup, щоб уникнути безіменних ідентифікаторів. Але інші дистрибутиви можуть трактувати цей ідентифікатор інакше, тому будь-яка портативна програма не повинна залежати від цього присвоєння.

Для деяких тестів у розділі 8 потрібен звичайний користувач. Ми додамо цього користувача тут і видалимо цей обліковий запис наприкінці цього розділу.

```
echo "tester:x:101:101::/home/tester:/bin/bash" >> /etc/passwd
echo "tester:x:101:" >> /etc/group
install -o tester -d /home/tester
```

Щоб прибрати запит «“I have no name!»», запустіть нову оболонку. Оскільки файли `/etc/passwd` і `/etc/group` було створено, визначення імен користувачів і груп тепер працюватиме:

```
exec /usr/bin/bash --login
```

Програми **login**, **agetty** та **init** (та інші) використовують ряд лог-файлів для запису інформації про те, хто і коли входив до системи. Однак, ці програми не будуть нічого записувати у файли журналів, якщо вони ще не існують. Ініціалізуйте файли журналів і надайте їм належні дозволи:

```
touch /var/log/{btmp,lastlog,faillog,wtmp}
chgrp -v utmp /var/log/lastlog
chmod -v 664 /var/log/lastlog
chmod -v 600 /var/log/btmp
```

Файл `/var/log/wtmp` записує всі входи і виходи з системи. Файл `/var/log/lastlog` записує дату, коли кожен користувач востаннє входив. Файл `/var/log/faillog` записує невдалі спроби входу. Файл `/var/log/btmp` записує помилкові спроби входу.



Note

У файлі `/run/utmp` записуються користувачі, які наразі увійшли до системи. Цей файл створюється динамічно у завантажувальних скриптах



Note

Файли `utmp`, `wtmp`, `btmp` та `lastlog` використовують 32-розрядні цілі числа для міток часу, і вони будуть фундаментально поламаними після 2038 року. Багато пакунків перестали їх використовувати і інші пакунки також перестануть їх використовувати. Напевно, краще вважати їх застарілими.

7.7. Gettext-0.24

Пакет Gettext містить утиліти для інтернаціоналізації та локалізації. Вони дозволяють компілювати програми з підтримкою NLS (Native Language Support), що дозволяє їм виводити повідомлення рідною мовою користувача.

Приблизний час побудови: 1.3 SBU

Необхідний простір на диску: 349 MB

7.7.1. Інсталяція Gettext

Для нашого тимчасового набору інструментів нам потрібно встановити лише три програми із Gettext.

Підготуйте Gettext до компіляції:

```
./configure --disable-shared
```

Значення параметрів конфігурації:

--disable-shared

Наразі нам не потрібно встановлювати жодної із спільних бібліотек Gettext, тому немає потреби їх збирати.

Зкомпілюйте пакет:

```
make
```

Встановіть **msgfmt**, **msgmerge** та **xgettext** програми:

```
cp -v gettext-tools/src/{msgfmt,msgmerge,xgettext} /usr/bin
```

Детальніше про цей пакунок наведено у Розділі 8.33.2, «Вміст Gettext».

7.8. Bison-3.8.2

До складу пакунка Bison входить генератор синтаксичного аналізатора.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 58 MB

7.8.1. Інсталяція Bison

Підготуйте Gettext до компіляції:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/bison-3.8.2
```

Значення нових параметрів конфігурації:

--docdir=/usr/share/doc/bison-3.8.2

Це вказує системі збірки встановити документацію bison до каталогу з версіями.

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make install
```

Детальніше про цей пакунок наведено у Розділі 8.34.2, «Вміст Bison».

7.9. Perl-5.40.1

Пакет Perl містить Practical Extraction та Report Language.

Приблизний час побудови: 0.6 SBU

Необхідний простір на диску: 285 MB

7.9.1. Інсталяція Perl

Підготуйте Perl до компіляції:

```
sh Configure -des \
-D prefix=/usr \
-D vendorprefix=/usr \
-D useshrplib \
-D privlib=/usr/lib/perl5/5.40/core_perl \
-D archlib=/usr/lib/perl5/5.40/core_perl \
-D sitelib=/usr/lib/perl5/5.40/site_perl \
-D sitearch=/usr/lib/perl5/5.40/site_perl \
-D vendorlib=/usr/lib/perl5/5.40/vendor_perl \
-D vendorarch=/usr/lib/perl5/5.40/vendor_perl
```

Значення параметрів конфігурації:

-des

Це комбінація трьох параметрів: *-d* використовує значення за замовчуванням для всіх елементів; *-e* забезпечує виконання всіх завдань; *-s* вимикає несуттєвий вивід.

-D vendorprefix=/usr

Це гарантує, що **perl** знає, як вказати пакункам, де вони мають встановити свої модулі Perl.

-D useshrplib

Збирає `libperl`, необхідний деяким модулям Perl, як спільну бібліотеку, а не як статичну.

-D privlib, -D archlib, -D sitelib, ...

Ці параметри визначають, де Perl шукатиме встановлені модулі. Редактори LFS вирішили розмістити їх у каталозі, заснованому на версії Perl MAJOR.MINOR (5.40), що дозволяє оновлювати Perl до новіших рівнів виправлень (рівень виправлення - це остання відокремлена крапкою частина у рядку повної версії, наприклад, 5.40.1) без перевстановлення всіх модулів.

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make install
```

Детальніше про цей пакунок наведено у Розділі 8.43.2, «Вміст Perl».

7.10. Python-3.13.2

Пакет Python 3 містить середовище розробки Python. Воно корисне для об'єктно-орієнтованого програмування, написання скриптів, створення прототипів великих програм і розробки додатків. Python - це інтерпретована комп'ютерна мова.

Приблизний час побудови: 0.5 SBU

Необхідний простір на диску: 634 MB

7.10.1. Інсталяція Python



Note

Є два архіви пакунків, назви яких починаються з префікса «python». Той, який потрібно розпакувати називається `Python-3.13.2.tar.xz` (зверніть увагу на велику першу літеру).

Підготуйте Python до компіляції:

```
./configure --prefix=/usr \
            --enable-shared \
            --without-ensurepip
```

Значення параметрів конфігурації:

`--enable-shared`

Цей параметр запобігає встановленню статичних бібліотек.

`--without-ensurepip`

Цей параметр вимикає програму встановлення пакунків Python, яка не потрібна на цьому етапі.

Зкомпілюйте пакет:

```
make
```



Note

Деякі модулі Python 3 не можуть бути зібрані зараз, оскільки залежності ще не встановлені. Для модуля `ssl`, виводиться повідомлення `Python requires a OpenSSL 1.1.1 or newer.` Це повідомлення слід проігнорувати. Просто переконайтесь, що команда `make` верхнього рівня не зазнала невдачі. Додаткові модулі зараз не потрібні, і їх буде зібрано у Розділі 8.

Встановіть пакет:

```
make install
```

Детальніше про цей пакунок наведено у Розділі 8.51.2, «Вміст Python».

7.11. Texinfo-7.2

Пакет Texinfo містить програми для читання, запису та перетворення інформаційних сторінок.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 152 MB

7.11.1. Інсталяція Texinfo

Підготуйте Texinfo до компіляції:

```
./configure --prefix=/usr
```

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make install
```

Детальніше про цей пакунок наведено у Розділі 8.72.2, «Вміст Texinfo».

7.12. Util-linux-2.40.4

Пакет Util-linux містить різноманітні утиліти.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 152 MB

7.12.1. Інсталяція Util-linux-2.40.4

FHS рекомендує використовувати каталог `/var/lib/hwclock` замість звичайного каталогу `/etc` для розміщення файлу `adjtime`. Створіть цей каталог за допомогою:

```
mkdir -pv /var/lib/hwclock
```

Підготуйте Util-linux-2.40.4 до компіляції:

```
./configure --libdir=/usr/lib \
            --runstatedir=/run \
            --disable-chfn-chsh \
            --disable-login \
            --disable-nologin \
            --disable-su \
            --disable-setpriv \
            --disable-runuser \
            --disable-pylibmount \
            --disable-static \
            --disable-liblastlog2 \
            --without-python \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.40.4
```

Значення параметрів конфігурації:

`ADJTIME_PATH=/var/lib/hwclock/adjtime`

Задає місце розташування файла з інформацією про апаратний годинник відповідно до FHS. Цей параметр не є строго необхідним для цього тимчасового інструменту, але він запобігає створенню файла в іншому місці, який не буде перезаписаний або видалений при складанні остаточного пакунка util-linux.

`--libdir=/usr/lib`

Цей параметр гарантує, що символічні посилання `.so` будуть спрямовані безпосередньо на файл спільної бібліотеки у тому ж каталозі (`/usr/lib`).

`--disable-*`

Ці параметри запобігають появи попереджень про збірку компонентів, які потребують пакунків, відсутніх у LFS або ще не встановлених.

`--without-python`

Цей параметр вимикає використання Python. Це дозволяє уникнути спроб зібрати непотрібні зв'язки.

`runstatedir=/run`

Цей параметр коректно встановлює розташування сокета, що використовується `uuidd` і `libuuid`.

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make install
```

Детальніше про цей пакунок наведено у Розділі 8.79.2, «Вміст Util-linux-2.40.4».

7.13. Очищення та збереження тимчасової системи

7.13.1. Очищення

По-перше, видаліть встановлені файли документації, щоб запобігти їх потраплянню у фінальну систему, також це збереже близько 35 МБ:

```
rm -rf /usr/share/{info,man,doc}/*
```

По-друге, у сучасній системі Linux файли libtool .la корисні лише для libltdl. Жодні бібліотеки у LFS не завантажуються за допомогою libltdl та відомо, що деякі файли .la можуть спричинити збої у роботі пакунків BLFS. Видаліть ці файли негайно:

```
find /usr/{lib,libexec} -name \*.la -delete
```

Поточний розмір системи становить близько 3 GB, проте каталог /tools більше не потрібен. Він займає близько 1 GB дискового простору. Видаліть його зараз:

```
rm -rf /tools
```

7.13.2. Backup

На цьому етапі основні програми та бібліотеки вже створені, і ваша поточна система LFS знаходиться в хорошому стані. Тепер можна створити резервну копію системи для подальшого використання. У разі фатальних збоїв у наступних розділах часто виявляється, що видалення всього і початок заново (більш обережно) є найкращим способом відновлення. На жаль, всі тимчасові файли також буде видалено. Щоб не витрачати зайвий час на переробку того, що вже було успішно зроблено, корисно створити резервну копію поточної системи LFS.



Note

Решта кроків у цьому розділі є необов'язковими. Проте, як тільки ви почнете встановлювати пакунки у Розділі 8, тимчасові файли буде перезаписано. Тому може бути корисним зробити резервну копію поточної системи, як описано нижче.

Наступні кроки виконуються ззовні середовища chroot. Це означає, що перш ніж продовжити, вам слід вийти з середовища chroot. Це необхідно для того, щоб отримати доступ до місць файлової системи за межами оточення chroot для зберігання/читання архіву резервної копії, який не повинен бути розміщений у межах ієрархії \$LFS.

Якщо ви вирішили створити резервну копію, вийдіть із середовища chroot:

```
exit
```



Important

Усі наведені нижче інструкції виконуються від імені користувача `root` у вашій хост-системі. Будьте особливо уважні до команд, які ви збираєтесь виконувати, оскільки помилки, допущені тут, можуть змінити вашу хост-систему. Майте на увазі, що змінна оточення `LFS` за замовчуванням встановлена для користувача `lfs`, але може бути не встановлена для користувача `root`.

Якщо ви хочете, щоб команди виконувалися від імені користувача `root`, переконайтесь, що ви встановили значення `LFS`.

Це було обговорено у Розділі 2.6, «Встановлення змінної `$LFS` та значення `umask`».

Перед створенням резервної копії відмонтуйте віртуальні файлові системи:

```
mountpoint -q $LFS/dev/shm && umount $LFS/dev/shm
umount $LFS/dev/pts
umount $LFS/{sys,proc,run,dev}
```

Переконайтесь, що у вас є щонайменше 1 GB вільного місця на диску (source tar-архіви буде включено до архіву резервної копії) у файловій системі, де міститься каталог, в який ви створюєте архів резервної копії.

Зверніть увагу, що у наведених нижче інструкціях вказано домашній каталог користувача `root`, який зазвичай знаходиться за адресою у кореневій файловій системі. Замініть `$HOME` на каталог на ваш вибір, якщо ви не хочете, щоб резервну копію було збережено у домашньому каталозі користувача `root`.

Створіть архів резервної копії, виконавши наступну команду:



Note

Оскільки архів резервної копії стискається, це займає відносно багато часу (понад 10 хвилин) навіть на досить швидкій системі

```
cd $LFS
tar -cJpf $HOME/lfs-temp-tools-12.3.tar.xz .
```



Note

Якщо ви продовжите у розділі 8, не забудьте знову увійти до середовища `chroot`, як описано у блокі «Important» нижче.

7.13.3. Відновлення

Якщо були допущені помилки і вам потрібно почати все спочатку, ви можете використати цю резервну копію для відновлення системи і заощадити час на відновлення. Оскільки вихідні коди знаходяться у файлі `$LFS`, вони також включені до архіву резервної копії, тому їх не потрібно завантажувати знову. Переконавшись, що `$LFS` налаштовано правильно, ви можете відновити систему видобувши архів резервної копії за допомогою наступних команд:



Warning

Наступні команди є надзвичайно небезпечними. Якщо ви запустите **rm -rf ./*** від імені користувача **root** і не зміните каталог на **\$LFS** або змінна оточення **LFS** не буде встановлена для користувача **root**, це знищить вашу хост-систему. **ВАС ПОПЕРЕДЖЕНО.**

```
cd $LFS
rm -rf ./*
tar -xpf $HOME/lfs-temp-tools-12.3.tar.xz
```

Ще раз перевірте, чи правильно налаштовано середовище, і продовжуйте збирати решту системи.



Important

Якщо ви вийшли з середовища **chroot** для створення резервної копії або перезапуску збірки за допомогою відновлення, не забудьте перевірити, що віртуальні файлові системи все ще змонтовано (**findmnt | grep \$LFS**). Якщо їх не змонтовано, змонтуйте їх зараз, як описано у Розділі 7.3, «Підготовка Virtual Kernel File Systems», і знову увійдіть до середовища **chroot** (див. Розділ 7.4, «Входження до середовища **chroot**»), перш ніж продовжити.

Частина IV. Збірка системи LFS

Глава 8. Встановлення базового системного програмного забезпечення

8.1. Вступ

У цьому розділі ми почнемо будувати систему LFS всерйоз.

Встановлення цього програмного забезпечення є простим. Хоча у багатьох випадках інструкції з встановлення можна було б зробити коротшими і загальнішими, ми вирішили надати повні інструкції для кожного пакунка, щоб звести до мінімуму можливість помилок. Ключ до розуміння роботи системи Linux - це знання того, для чого використовується кожен пакунок і чому він може знадобитися вам (або системі).

Ми не рекомендуємо використовувати кастомні оптимізації. Вони можуть дещо пришвидшити роботу програми, але можуть також спричинити труднощі під час компіляції та проблеми під час запуску програми. Якщо пакунок відмовляється компілюватися зі спеціальною оптимізацією, спробуйте скомпілювати його без оптимізації і перевірте, чи це вирішить проблему. Навіть якщо пакунок компілюється при використанні налаштованої оптимізації, існує ризик, що його було скомпільовано неправильно через складну взаємодію між кодом та інструментами збірки. Також зауважте, що опції `-march` і `-mtune`, які використовують значення, не вказані у книзі, не було протестовано. Це може спричинити проблеми з пакунками інструментарію (Binutils, GCC та Glibc). Невеликий потенційний вигравш, досягнутий завдяки налаштуванню оптимізації компілятора, часто переважає ризики. Початківцям, які збирають LFS, рекомендується збирати без спеціальних оптимізацій.

З іншого боку, ми зберігаємо оптимізації, увімкнені у конфігурації пакунків за замовчуванням. Крім того, ми іноді явно вмикаємо оптимізовану конфігурацію, яка надається пакунком, але не ввімкнена за замовчуванням. Розробники пакунка вже протестували ці конфігурації і вважають їх безпечними, тож малоймовірно, що вони порушать роботу збірки. Зазвичай конфігурація за замовчуванням вже вмикає `-O2` або `-O3`, тож отримана система працюватиме дуже швидко без жодної оптимізації, і водночас буде стабільною.

Перед інструкціями зі встановлення на кожній сторінці встановлення надається інформація про пакунок, зокрема стислий опис того, що він містить, приблизний час збирання та обсяг дискового простору, необхідний під час цього процесу. Після інструкцій зі встановлення наведено список програм і бібліотек (разом з коротким описом), які пакунок встановлює.

Note

Значення SBU та необхідного дискового простору включають дані тестових наборів для всіх застосовних пакетів у Главі 8. Значення SBU розраховано з використанням чотирьох ядер процесора (`-j4`) для всіх операцій, якщо не вказано інше.

8.1.1. Про бібліотеки

Загалом, редактори LFS не рекомендують створювати та встановлювати статичні бібліотеки. Більшість статичних бібліотек стали застарілими у сучасній системі Linux. Крім того, підключення статичної бібліотеки до програми може бути шкідливим. Якщо для усунення проблеми безпеки необхідно оновити бібліотеку,

кожну програму, яка використовує статичну бібліотеку, потрібно буде перекомпонувати з новою бібліотекою. Оскільки використання статичних бібліотек не завжди є очевидним, відповідні програми (і процедури, необхідні для компонування) можуть навіть не бути відомими.

Процедури, описані у цьому розділі, вилучають або вимикають встановлення більшості статичних бібліотек. Зазвичай це робиться за допомогою параметра `--disable-static` в **configure**. В інших випадках вам знадобляться інші засоби. У деяких випадках, зокрема у Glibc та GCC, використання статичних бібліотек залишається невід'ємною частиною процесу збирання пакунків.

Докладніше про бібліотеки можна дізнатися на сторінці Бібліотеки: Статичні чи динамічні? у книзі BLFS.

8.2. Керування пакетами

Керування пакунками є часто запитуваним доповненням до книги LFS. Менеджер пакунків відстежує встановлення файлів, що полегшує видалення та оновлення пакунків. Хороший менеджер пакунків також обробляє конфігураційні файли спеціально для того, щоб зберегти конфігурацію користувача, коли пакунок перевстановлюється або оновлюється. Перш ніж ви почнете дивуватися, НІ - у цьому розділі не йдеться про жодний конкретний менеджер пакунків і не рекомендується жодного з них. Натомість у ньому наведено огляд найпопулярніших методів і принципів їхньої роботи. Ідеальний для вас менеджер пакунків може бути серед цих методів, або це може бути комбінація двох чи більше з них. У цьому розділі коротко згадано проблемами, які можуть виникнути під час оновлення пакунків.

Деякі причини, чому жоден менеджер пакунків не згадується у LFS або BLFS, є такими:

- Розгляд управління пакунками відволікає увагу від мети цих книг - навчити, як побудована система Linux.
- Існує безліч рішень для управління пакетами, кожне з яких має свої переваги та недоліки. Знайти одне рішення, яке б задовольнило всі аудиторії, досить складно.

Існує кілька підказок на тему керування пакунками. Відвідайте *Hints Project* і подивіться, чи одна з них відповідає вашим потребам.

8.2.1. Проблеми з оновленням

Менеджер пакунків дозволяє легко оновити пакунки до нових версій, коли вони виходять. Як правило, для оновлення до нових версій можна скористатися інструкціями з книг LFS і BLFS на сайті. Нижче наведено деякі моменти, на які слід звернути увагу при оновленні пакунків, особливо на працючій системі.

- Якщо потрібно оновити ядро Linux (наприклад, з 5.10.17 до 5.10.18 або 5.11.1), більше нічого не потрібно переробляти. Система продовжить нормально працювати завдяки чітко визначеному інтерфейсу між ядром і користувачьким простором. Зокрема, заголовки Linux API не потрібно оновлювати разом з ядром. Вам потрібно лише перезавантажити систему, щоб використовувати оновлене ядро.
- Якщо Glibc потрібно оновити до новішої версії (наприклад, з Glibc-2.36 до Glibc-2.41), необхідно виконати деякі додаткові кроки, щоб уникнути поломки системи. Докладні відомості наведено у Розділі 8.5, «Glibc-2.41».
- Якщо пакунок, що містить спільну бібліотеку, оновлено, і якщо назву бібліотеки змінено, то всі пакунки, динамічно з'єднані з бібліотекою, необхідно перекомпілювати, щоб з'єднати їх з новою бібліотекою. (Зауважте, що між версією пакунка і назвою бібліотеки не існує кореляції). Наприклад, розглянемо пакунок

foo-1.2.3, який встановлює динамічну бібліотеку з назвою `libfoo.so.1`. Припустімо, що ви оновили пакунок до новішої версії foo-1.2.4, яка встановлює динамічну бібліотеку з іменем `libfoo.so.2`. У цьому випадку всі пакунки, які динамічно пов'язані з `libfoo.so.1`, потрібно перекомпілювати для посилання на `libfoo.so.2`, щоб використовувати нову версію бібліотеки. Не слід видаляти старі бібліотеки, доки не буде перекомпоновано всі залежні від них пакунки.

- Якщо пакунок (прямо чи опосередковано) пов'язано як зі старим, так і з новим іменами динамічної бібліотеки (наприклад, пакунок посилається як на `libfoo.so.2`, так і на `libbar.so.1`, тоді як останній посилається на `libfoo.so.3`), пакунок може не працювати, оскільки різні версії динамічної бібліотеки містять несумісні визначення для деяких символів імен. Це може бути спричинено перекомпіляцією деяких, але не всіх, пакунків, пов'язаних зі старою динамічною бібліотекою, після оновлення пакунка, який надає динамічну бібліотеку. Щоб уникнути цієї проблеми, користувачам слід якнайшвидше перекомпілювати кожен пакунок, пов'язаний з бібліотекою, до оновленої ревізії (наприклад, `libfoo.so.2` до `libfoo.so.3`).
- Якщо пакунок, що містить спільну бібліотеку, оновлюється, і назва бібліотеки не змінюється, але зменшується номер версії файлу бібліотеки (наприклад, бібліотека все ще має назву `libfoo.so.1`, але ім'я файлу бібліотеки змінюється з `libfoo.so.1.25` на `libfoo.so.1.24`), вам слід видалити файл бібліотеки з попереднього пакунка встановлену версію (у цьому випадку `libfoo.so.1.25`). Інакше команда **ldconfig** (викликана вами з командного рядка або під час встановлення якогось пакунка) скине символічне посилання `libfoo.so.1` і вкаже на старий файл бібліотеки, оскільки він видасться «новішою» версією, оскільки його номер версії більший. Така ситуація може виникнути, якщо вам потрібно понизити версію пакунка, або якщо автори змінять схему версійності бібліотечних файлів.
- Якщо пакунок, що містить спільну бібліотеку, оновлюється, і назва бібліотеки не змінюється, але виправляється серйозна проблема (зокрема, уразливість у безпеці), всі запущені програми, пов'язані з спільною бібліотекою, слід перезапустити. Наступна команда, виконана від імені користувача `root` після завершення оновлення, покаже, які процеси використовують стари версії цих бібліотек (замініть `libfoo` на назву бібліотеки):

```
grep -l 'libfoo.*deleted' /proc/*maps | tr -cd 0-9\n | xargs -r ps u
```

Якщо для доступу до системи використовується OpenSSH і він пов'язаний з оновленою бібліотекою, необхідно перезапустити службу **sshd**, потім вийти з системи, увійти знову і знову виконати попередню команду, щоб переконатися, що ніщо не використовує видалені бібліотеки.

- Якщо виконувану програму або спільну бібліотеку перезаписано, процеси, що використовують код або дані цієї програми або бібліотеки, можуть завершитися аварійно. Правильний спосіб оновити програму або бібліотеку, до якої надається динамічний доступ, не спричинивши при цьому аварійного завершення процесу - це спочатку видалити її, а потім встановити нову версію. Команда **install** з coreutils вже реалізувала це, і більшість пакунків використовують цю команду для встановлення двійкових файлів і бібліотек. Це означає, що вас не буде турбувати ця проблема більшу частину часу. Однак процес встановлення деяких пакунків (зокрема, SpiderMonkey у BLFS) просто перезаписує файл, якщо він існує; це призводить до аварійного завершення роботи. Тому безпечно зберегти вашу роботу і закрити непотрібні запущені процеси перед оновленням пакунка.

8.2.2. Техніки керування пакетами

Нижче наведено кілька поширеніх методів керування пакунками. Перш ніж прийняти рішення про вибір менеджера пакунків, знайдіть інформацію про різні методи, зокрема, про недоліки кожної конкретної схеми.

8.2.2.1. Це все в моїй голові!

Так, це метод керування пакунками. Деяким користувачам не потрібен менеджер пакунків, оскільки вони добре знають пакунки і знають, які файли встановлюються кожним пакунком. Деяким користувачам також не потрібне керування пакунками, оскільки вони планують перебирати всю систему щоразу, коли змінюється пакунок.

8.2.2.2. Встановлювати в окремі директорії

Це простий спосіб керування пакунками, який не потребує спеціальної програми для керування пакунками. Кожен пакунок встановлюється в окремий каталог. Наприклад, пакунок foo-1.1 встановлено у /opt/foo-1.1, а з /opt/foo до /opt/foo-1.1 створено символічне посилання. Коли з'являється нова версія foo-1.2, вона встановлюється у /opt/foo-1.2 і попереднє посилання замінюється на посилання на нову версію.

Змінні оточення, такі як PATH, MANPATH, INFOPATH, PKG_CONFIG_PATH, CPPFLAGS, LDFLAGS та конфігураційний файл /etc/ld.so.conf, можливо, потрібно буде розширити, щоб включити відповідні підкаталоги в /opt/foo-x.у.

Ця схема використовується у книзі BLFS для встановлення деяких дуже великих пакунків, щоб полегшити їхнє оновлення. Якщо ви встановлюєте більше кількох пакунків, ця схема стає некерованою. А деякі пакунки (наприклад, заголовки Linux API та Glibc) можуть не працювати за цією схемою. **Ніколи не використовуйте цю схему для всієї системи.**

8.2.2.3. Керування пакетами в стилі символічних посилань

Це варіація попереднього способу керування пакунками. Кожен пакунок встановлюється так само, як і у попередній схемі. Але замість того, щоб створювати символічне посилання на загальну назву пакунка, кожен файл символічно прив'язується до ієрархії /usr. Це усуває необхідність розширювати змінні оточення. Хоча символічні посилання може створювати користувач, багато менеджерів пакунків використовують цей підхід і автоматизують створення символічних посилань. Деякі з популярних програм включають Stow, Epkg, Graft і Depot.

Скрипт встановлення потрібно обдурити, щоб пакунок думав, що його встановлено у /usr, хоча насправді його встановлено у ієрархії /usr/pkg. Встановлення у такий спосіб зазвичай не є тривіальним завданням. Наприклад, припустімо, що ви встановлюєте пакунок libfoo-1.1 з. За наведеними нижче інструкціями пакунок може бути встановлено неправильно:

```
./configure --prefix=/usr/pkg/libfoo/1.1
make
make install
```

Встановлення відбудеться, але залежні пакунки можуть не з'єднатися з libfoo так, як ви очікували. Якщо ви скомпілюєте пакунок, який посилається на libfoo, ви можете помітити, що він посилається на /usr/pkg/libfoo/1.1/lib/libfoo.so.1, а не на /usr/lib/libfoo.so.1, як ви могли б очікувати.

Правильним підходом є використання змінної DESTDIR для керування встановленням. Цей підхід працює наступним чином:

```
./configure --prefix=/usr
make
make DESTDIR=/usr/pkg/libfoo/1.1 install
```

Більшість пакунків підтримують цей підхід, але є й такі, що не підтримують. Для пакунків, які не підтримують цей підхід, вам може знадобитися встановити пакунок вручну, або ви можете виявити, що деякі проблемні пакунки простіше встановити у /opt.

8.2.2.4. На основі часових міток

У цьому методі файл позначається часовою міткою перед встановленням пакунка. Після встановлення просте використання команди **find** з відповідними параметрами може згенерувати журнал усіх файлів, встановлених після того, як було створено файл з часовою міткою. Менеджер пакунків, який використовує такий підхід, - **install-log**.

Хоча перевагою цієї схеми є простота, вона має два недоліки. Якщо під час встановлення пакунків встановлено з будь-якою міткою часу, відмінною від поточного, ці пакунки не буде відстежено менеджером пакунків. Крім того, цю схему можна використовувати лише тоді, коли пакунки встановлюються по одному. Журнали не є надійними, якщо два пакунки встановлюються одночасно з двох різних консолей.

8.2.2.5. Трасування інсталяційних скриптів

У цьому підході записуються команди, які виконують інсталяційні скрипти. Існує два способи, ось один що можна використовувати:

Змінна середовища **LD_PRELOAD** може вказувати на бібліотеку, яку слід попередньо завантажити перед встановленням. Під час встановлення ця бібліотека відстежує пакунки, які встановлюються, приєднуючись до різних виконуваних файлів, таких як **cp**, **install**, **mv** і відстежуючи системні виклики, які модифікують файлову систему. Для того, щоб цей підхід працював, всі виконувані файли повинні бути динамічно злінковані без бітів **suid** або **sgid**. Попереднє завантаження бібліотеки може спричинити деякі небажані побічні ефекти під час встановлення. Тому варто виконати кілька тестів, щоб переконатися, що менеджер пакунків нічого не зламає і що він реєструє всі відповідні файли.

Іншим методом є використання **strace**, який записує всі системні виклики, зроблені під час виконання інсталяційних скриптів.

8.2.2.6. Створення архівів пакунків

За такою схемою встановлення пакунка імітується в окреме дерево, як було описано раніше в розділі, присвяченому управлінню пакунками за допомогою символічних посилань. Після встановлення, використовуючи встановлені файли, створюється архів пакунка. Цей архів потім використовується для встановлення пакунка на локальній машині або навіть на інших машинах.

Цей підхід використовується більшістю менеджерів пакунків у комерційних дистрибутивах. Прикладами менеджерів пакунків, які дотримуються цього підходу, є **RPM** (який, до речі, вимагається Специфікацією стандартної бази Linux), **pkg-utils**, **apt** у **Debian** та система **Portage** у **Gentoo**. Підказка, яка описує, як

прийняти цей стиль керування пакунками для систем LFS, знаходиться за адресою <https://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt>.

Створення файлів пакунків, які містять інформацію про залежності, є складним завданням і виходить за рамки LFS.

Slackware використовує систему на основі **tar** для архівів пакунків. Ця система навмисно не обробляє залежності пакунків, як це роблять більш складні менеджери пакунків. Докладні відомості про керування пакунками у Slackware наведено на сторінці <https://www.slackbook.org/html/package-management.html>.

8.2.2.7. Управління на основі користувачів

Цю унікальну для LFS схему розробив Матіас Бенкманн, і її можна знайти у *Hints Project*. У цій схемі кожен пакунок встановлюється окремим користувачем у стандартні місця. Файли, що належать до пакунка, легко ідентифікувати, перевіривши ідентифікатор користувача. Можливості та недоліки цього підходу є надто складними, щоб описати їх у цьому розділі. Для отримання детальної інформації зверніться до підказки на https://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt.

8.2.3. Розгортання LFS на декількох системах

Однією з переваг системи LFS є те, що у ній немає файлів, які залежать від розташування файлів на диску. Клонувати збірку LFS на інший комп'ютер з такою ж архітектурою, як і базова система, так само просто, як і використовувати **tar** на розділі LFS, який містить кореневий каталог (блізько 900 МВ без стиснення для базової збірки LFS), скопіювати цей файл через мережу або CD-ROM / USB-накопичувач до нової системи і розгорнути його. Після цього потрібно буде змінити кілька конфігураційних файлів. Файли конфігурації, які, можливо, потрібно буде оновити, включають: `/etc/hosts`, `/etc/fstab`, `/etc/passwd`, `/etc/group`, `/etc/shadow`, `/etc/ld.so.conf`, `/etc/sysconfig/rc.site`, `/etc/sysconfig/network` і `/etc/sysconfig/ifconfig.eth0`.

Для нової системи може знадобитися спеціальне ядро, залежно від відмінностей у апаратному забезпеченні системи та оригінальній конфігурації ядра.



Note

Були повідомлення про проблеми під час копіювання між схожими, але не ідентичними архітектурами. Наприклад, набір інструкцій для системи Intel не ідентичний інструкціям процесора AMD, а пізніші версії деяких процесорів можуть містити інструкції, недоступні в попередніх версіях.

Нарешті, нову систему потрібно зробити завантаженою за допомогою Розділу 10.4, «Використання GRUB для налаштування процесу завантаження».

8.3. Man-pages-6.12

Пакет Man-pages містить понад 2400 довідкових сторінок тан.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 52 MB

8.3.1. Інсталяція Man-pages

Видалити дві сторінки тан для функцій хешування паролів. Libxcrypt надасть кращу версію цих сторінок man:

```
rm -v man3/crypt*
```

Встановіть Man-pages запустивши:

```
make -R GIT=false prefix=/usr install
```

Значення параметрів:

-R

Це запобігає встановленню **make** будь-яких вбудованих змінних. Система побудови тан-сторінок не працює добре з вбудованими змінними, але наразі немає способу їх вимкнути, окрім як явно передати **-R** через командний рядок.

GIT=false

Це запобігає виведенню системою побудови багатьох рядків попередження git: command not found.

8.3.2. Вміст Man-pages

Встановлені файли: різні сторінки тан

Короткі описи

man pages

Описує функції мови програмування C, важливі файли пристроїв та основні файли конфігурації.

8.4. lana-Etc-20250123

Пакет lana-Etc надає дані для мережевих служб і протоколів.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 4.8 MB

8.4.1. Інсталяція lana-Etc

Для цього пакета нам потрібно лише скопіювати файли в потрібне місце:

```
cp services protocols /etc
```

8.4.2. Вміст lana-Etc

Встановлені файли: /etc/protocols та /etc/services

Короткі описи

/etc/protocols	Описує різні протоколи DARPA Internet, які доступні з підсистеми TCP/IP.
/etc/services	Забезпечує відповідність між зручними текстовими назвами інтернет-сервісів та їхніми базовими призначеними номерами портів і типами протоколів.

8.5. Glibc-2.41

Пакет Glibc містить основну бібліотеку С. Ця бібліотека надає основні процедури для виділення пам'яті, пошуку каталогів, відкриття та закриття файлів, читання та запису файлів, обробки рядків, зіставлення зразків, арифметики тощо.

Приблизний час побудови: 12 SBU

Необхідний простір на диску: 3.2 GB

8.5.1. Інсталяція Glibc

Деякі програми Glibc використовують каталог `/var/db`, який не відповідає FHS, для зберігання своїх даних виконання. Застосуйте наступний патч, щоб такі програми зберігали свої дані виконання в місцях, що відповідають FHS:

```
patch -Np1 -i ../glibc-2.41-fhs-1.patch
```

Документація Glibc рекомендує компілювати Glibc у окремому каталозі `build`:

```
mkdir -v build
cd      build
```

Переконайтесь, що утиліти `ldconfig` та `sln` будуть встановлені в `/usr/sbin`:

```
echo "rootsbindir=/usr/sbin" > configparms
```

Підготуйте Glibc до компіляції:

```
./configure --prefix=/usr \
            --disable-werror \
            --enable-kernel=5.4 \
            --enable-stack-protector=strong \
            --disable-nscd \
            libc_cv_slibdir=/usr/lib
```

Значення конфігураційних параметрів:

`--disable-werror`

Ця опція вимикає опцію `-Werror`, передану GCC. Це необхідно для запуску набору тестів.

`--enable-kernel=5.4`

Ця опція повідомляє системі збірки, що ця Glibc може використовуватися з ядрами версії 5.4 і старше. Це означає створення обхідних рішень на випадок, якщо системний виклик, введений в пізнішій версії, не може бути використаний.

`--enable-stack-protector=strong`

Ця опція підвищує безпеку системи, додаючи додатковий код для перевірки переповнення буфера, наприклад, атак типу `stack smashing`. Зверніть увагу, що Glibc завжди явно замінює значення за замовчуванням GCC, тому ця опція все ще потрібна, навіть якщо ми вже вказали `--enable-default-ssp` для GCC.

`--disable-nscd`

Не створювати демон кешу служб імен, який більше не використовується.

`libc_cv_slibdir=/usr/lib`

Ця змінна встановлює правильну бібліотеку для всіх систем. Ми не хочемо, щоб використовувалася `lib64`.

Зкомпілюйте пакет:

```
make
```

Important

У цьому розділі набір тестів для Glibc вважається критично важливим. Не пропускайте його за жодних обставин.

Зазвичай кілька тестів не проходять. Перелічені нижче невдалі тести зазвичай можна безпечно ігнорувати.

```
make check
```

Ви можете побачити деякі невдалі тести. Набір тестів Glibc дещо залежить від хост-системи. Кілька невдалих тестів із понад 6000 тестів зазвичай можна ігнорувати. Ось список найпоширеніших проблем, що зустрічаються в останніх версіях LFS:

- Відомо, що *io/tst-lchmod* не працює в середовищі LFS chroot.
- Деякі тести, наприклад *nss/tst-nss-files-hosts-multi* та *nptl/tst-thread-affinity**, як відомо, завершуються з помилкою через перевищення часу очікування (особливо коли система працює відносно повільно та/або тестовий набір виконується з декількома паралельними завданнями make). Ці тести можна ідентифікувати за допомогою:

```
grep "Timed out" $(find -name *.out)
```

Можна повторно запустити окремий тест із збільшеним часом очікування за допомогою **TIMEOUTFACTOR=<factor>** **make test t=<test-name>**. Наприклад, **TIMEOUTFACTOR=10 make testt=nss/tst-nss-files-hosts-multi** повторно запустить *nss/tst-nss-files-hosts-multi* із десятикратним збільшенням початкового часу очікування.

Крім того, деякі тести можуть завершитися невдачею на відносно старих моделях процесорів (наприклад, *elf/tst-cpu-features-cpuinfo*) або версіях ядра хоста (наприклад, *stdlib/tst-arc4random-thread*).

Хоча це нешкідливе повідомлення, на етапі встановлення Glibc з'явиться повідомлення про відсутність файлу */etc/ld.so.conf*. Запобігти цьому попередженню можна за допомогою:

```
touch /etc/ld.so.conf
```

Виправте Makefile, щоб пропустити застарілу перевірку працевздатності, яка не проходить у сучасній конфігурації Glibc:

```
sed '/test-installation/s@$(PERL) @echo not running@' -i ./Makefile
```



Important

Якщо ви оновлюєте Glibc до нової мінорної версії (наприклад, з Glibc-2.36 до Glibc-2.41) на працючій системі LFS, вам потрібно вжити додаткових запобіжних заходів, щоб уникнути пошкодження системи:

- Оновлення Glibc на системі LFS до версії 11.0 (ексклюзивно) не підтримується. Перекомпілюйте LFS, якщо ви використовуєте таку стару систему LFS, але вам потрібна новіша версія Glibc.
- Якщо ви оновлюєте систему LFS до версії 12.0 (виключно), встановіть Libxcrypt відповідно до розділу 8.27, «Libxcrypt-4.4.38». На додаток до звичайної інсталяції Libxcrypt, **ви ПОВИННІ дотримуватися вказівки в розділі Libxcrypt, щоб інсталювати libcrypt.so.1*** (замість libcrypt.so.1 з попередньої інсталяції Glibc).
- Якщо ви оновлюєте систему LFS до версії 12.1 (виключно), видаліть програму **nscd**:

```
rm -f /usr/sbin/nscd
```

- Оновіть ядро і перезавантажте систему, якщо версія старша за 5.4 (перевірте поточну версію за допомогою команди **uname -r**) або якщо ви хочете оновити її в будь-якому випадку, дотримуйтесь вказівок розділу 10.3, «Linux-6.13.4».
- Оновіть заголовки API ядра, якщо вони старші за 5.4 (перевірте поточну версію за допомогою **cat /usr/include/linux/version.h**) або якщо ви все одно хочете їх оновити, дотримуючись вказівок розділу 5.4, «Заголовки API Linux-6.13.4» (але видаливши \$LFS з команди **cp**).
- Виконайте інсталяцію **DESTDIR** та оновлення спільних бібліотек Glibc у системі за допомогою однієї єдиної команди **install**:

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/*.so.* /usr/lib
```

Необхідно суворо дотримуватися наведених вище кроків, якщо ви не розумієте, що саме робите. **Будь-яке несподіване відхилення може привести до повної непрацездатності системи. ВИ ПОПЕРЕДЖЕНІ.**

Потім продовжуйте виконувати команду **make install**, команду **sed** для **/usr/bin/ldd** та команди для встановлення локалей. Після їх виконання негайно перезавантажте систему.

Після успішного перезавантаження системи, якщо ви використовуєте систему LFS версії нижче 12.0 (виключно), в якій GCC не було скомпільовано з опцією **--disable-fixincludes**, перемістіть два заголовки GCC в більш підходяще місце та видаліть застарілі «віправлені» копії заголовків Glibc:

```
DIR=$(dirname $(gcc -print-libgcc-file-name))
[ -e $DIR/include/limits.h ] || mv $DIR/include{-fixed,}/limits.h
[ -e $DIR/include/syslimits.h ] || mv $DIR/include{-fixed,}/syslimits.h
rm -rfv $(dirname $(gcc -print-libgcc-file-name))/include-fixed/*
```

Встановіть пакет:

```
make install
```

Виправте жорстко заданий шлях до виконуваного завантажувача в скрипті ldd:

```
sed '/RTLDLIST=/s@/usr@@g' -i /usr/bin/ldd
```

Далі встановіть локалі, які дозволяють системі відповідати іншою мовою. Жодна з цих локалей не є обов'язковою, але якщо деякі з них відсутні, набори тестів деяких пакетів пропускатимуть важливі тестові випадки.

Окремі локалі можна встановити за допомогою програми **localeddef**. Наприклад, друга команда **localeddef** нижче поєднує незалежне від кодування символів визначення локалі `/usr/share/i18n/locales/cs_CZ 3 /usr/share/i18n/charmaps=UTF-8. gz` і додає результат до файлу `/usr/lib/locale/locale-archive`. Наступні інструкції встановлять мінімальний набір локалей, необхідний для оптимального охоплення тестів:

```
localedef -i C -f UTF-8 C.UTF-8
localedef -i cs_CZ -f UTF-8 cs_CZ.UTF-8
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i de_DE -f UTF-8 de_DE.UTF-8
localedef -i el_GR -f ISO-8859-7 el_GR
localedef -i en_GB -f ISO-8859-1 en_GB
localedef -i en_GB -f UTF-8 en_GB.UTF-8
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_ES -f ISO-8859-15 es_ES@euro
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i is_IS -f ISO-8859-1 is_IS
localedef -i is_IS -f UTF-8 is_IS.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i it_IT -f ISO-8859-15 it_IT@euro
localedef -i it_IT -f UTF-8 it_IT.UTF-8
localedef -i ja_JP -f EUC-JP ja_JP
localedef -i ja_JP -f SHIFT_JIS ja_JP.SJIS 2> /dev/null || true
localedef -i ja_JP -f UTF-8 ja_JP.UTF-8
localedef -i nl_NL@euro -f ISO-8859-15 nl_NL@euro
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i se_NO -f UTF-8 se_NO.UTF-8
localedef -i ta_IN -f UTF-8 ta_IN.UTF-8
localedef -i tr_TR -f UTF-8 tr_TR.UTF-8
localedef -i zh_CN -f GB18030 zh_CN.GB18030
localedef -i zh_HK -f BIG5-HKSCS zh_HK.BIG5-HKSCS
localedef -i zh_TW -f UTF-8 zh_TW.UTF-8
```

Крім того, встановіть локаль для своєї країни, мови та набору символів.

Або ж встановіть усі локалі, перелічені у файлі `glibc-2.41/localedata/SUPPORTED` (він містить усі локалі, перелічені вище, та багато інших) за допомогою наступної команди, що займає багато часу:

```
make localesdata/install-locales
```

Потім використовуйте команду `localeddef` для створення та встановлення локалей, які не вказані у файлі `glibc-2.41/localedata/SUPPORTED`, коли вони вам знадобляться. Наприклад, для деяких тестів, що будуть наведені далі в цьому розділі, потрібні дві наступні локалі:

```
localedef -i C -f UTF-8 C.UTF-8
localedef -i ja_JP -f SHIFT_JIS ja_JP.SJIS 2> /dev/null || true
```



Note

Glibc тепер використовує `libidn2` для вирішення інтернаціоналізованих доменних імен. Це залежність від часу виконання. Якщо ця функція необхідна, інструкції з інсталяції `libidn2` можна знайти на сторінці *BLFS libidn2*.

8.5.2. Конфігурація Glibc

8.5.2.1. Додавання nsswitch.conf

Файл `/etc/nsswitch.conf` потрібно створити, оскільки стандартні налаштування Glibc не працюють належним чином у мережевому середовищі.

Створіть новий файл `/etc/nsswitch.conf`, виконавши наступну команду:

```
cat > /etc/nsswitch.conf << "EOF"
# Begin /etc/nsswitch.conf

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

protocols: files
services: files
ethers: files
rpc: files

# End /etc/nsswitch.conf
EOF
```

8.5.2.2. Додавання даних про часовий пояс

Встановіть і налаштуйте дані часового поясу за допомогою наступних команд:

```
tar -xf ../../tzdata2025a.tar.gz

ZONEINFO=/usr/share/zoneinfo
mkdir -pv $ZONEINFO/{posix,right}

for tz in etcetera southamerica northamerica europe africa antarctica \
          asia australasia backward; do
    zic -L /dev/null -d $ZONEINFO ${tz}
    zic -L /dev/null -d $ZONEINFO/posix ${tz}
    zic -L leapseconds -d $ZONEINFO/right ${tz}
done

cp -v zone.tab zone1970.tab iso3166.tab $ZONEINFO
zic -d $ZONEINFO -p America/New_York
unset ZONEINFO tz
```

Значення зіс команд:`zic -L /dev/null ...`

Це створює часові пояси POSIX без додаткових секунд. Зазвичай їх розміщують як у `zoneinfo`, так і в `zoneinfo/posix`. Необхідно розмістити часові пояси POSIX у `zoneinfo`, інакше різні набори тестів повідомлятимуть про помилки. На вбудованій системі, де простір обмежений і ви не плануєте оновлювати часові пояси, ви можете заощадити 1,9 МБ, не використовуючи каталог `posix`, але деякі програми або набори тестів можуть видавати помилки.

`zic -L leapseconds ...`

Це створює правильні часові пояси, включаючи додаткові секунди. У вбудованій системі, де місце обмежене і ви не плануєте оновлювати часові пояси або не дбаєте про правильний час, ви можете заощадити 1,9 МБ, опустивши `right` каталог.

`zic ... -p ...`

Це створює файл `posixrules`. Ми використовуємо Нью-Йорк, оскільки POSIX вимагає, щоб правила переходу на літній час відповідали правилам США.

Один із способів визначити місцевий часовий пояс — запустити такий скрипт:

tzselect

Після відповіді на кілька запитань про місцезнаходження, скрипт виведе назву часового поясу (наприклад, `America/Edmonton`). У файлі `/usr/share/zoneinfo` також перелічені деякі інші можливі часові пояси, такі як `Canada/Eastern` або `EST5EDT`, які не ідентифікуються скриптом, але можуть бути використані.

Потім створіть файл `/etc/localtime`, виконавши:

```
ln -sfv /usr/share/zoneinfo/<xxx> /etc/localtime
```

Замініть `<xxx>` на назву вибраного часового поясу (наприклад, `Canada/Eastern`).

8.5.2.3. Налаштування динамічного завантажувача

За замовчуванням динамічний завантажувач (`/lib/ld-linux.so.2`) шукає в `/usr/lib` динамічні бібліотеки, які потрібні програмам під час їх виконання. Однак, якщо бібліотеки знаходяться в інших каталогах, крім `/usr/lib`, їх потрібно додати до файлу `/etc/ld.so.conf`, щоб динамічний завантажувач міг

їх знайти. Два каталоги, які, як відомо, містять додаткові бібліотеки, — це `/usr/local/lib` та `/opt/lib`, тому додайте ці каталоги до шляху пошуку динамічного завантажувача.

Створіть новий файл `/etc/ld.so.conf`, виконавши наступне:

```
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf
/usr/local/lib
/opt/lib

EOF
```

За бажанням динамічний завантажувач також може шукати в каталозі та включати вміст знайдених там файлів. Зазвичай файли в цьому каталозі містять один рядок, що вказує бажаний шлях до бібліотеки. Щоб додати цю функцію, виконайте наступні команди:

```
cat >> /etc/ld.so.conf << "EOF"
# Add an include directory
include /etc/ld.so.conf.d/*.conf

EOF
mkdir -pv /etc/ld.so.conf.d
```

8.5.3. Вміст Glibc

Встановлені програми:	gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, ld.so (символічне посилання на ld-linux-x86-64.so.2 або ld-linux.so.2), locale, localedef, makedb, mtrace, pcprofiledump, pldd, sln, sotruss, sprof, tzselect, xtrace, zdump, and zic
Встановлені бібліотеки:	ld-linux-x86-64.so.2, ld-linux.so.2, libBrokenLocale.{a,so}, libanl.{a,so}, libc.{a,so}, libc_nonshared.a, libc_malloc_debug.so, libdl.{a,so.2}, libg.a, libm.{a,so}, libmcheck.a, libmemusage.so, libmvec.{a,so}, libnsl.so.1, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss hesiod.so, libpcprofile.so, libpthread.{a,so.0}, libresolv.{a,so}, librt.{a,so.1}, libthread_db.so, and libutil.{a,so.1}
Встановлені директорії:	/usr/include/arpa, /usr/include/bits, /usr/include/gnu, /usr/include/net, /usr/include/netash, /usr/include/netatalk, /usr/include/netax25, /usr/include/neteconet, /usr/include/netinet, /usr/include/netipx, /usr/include/netiucv, /usr/include/netpacket, /usr/include/netrom, /usr/include/netrose, /usr/include/nfs, /usr/include/protocols, /usr/include/rpc, /usr/include/sys, /usr/lib/audit, /usr/lib/gconv, /usr/lib/locale, /usr/libexec/getconf, /usr/share/i18n, /usr/share/zoneinfo, та /var/lib/nss_db

Короткі описи

gencat

Генерує каталоги повідомлень.

getconf

Відображає значення конфігурації системи для змінних, специфічних для файлової системи.

getent	Отримує записи з адміністративної бази даних.
iconv	Виконує перетворення набору символів.
iconvconfig	Створює файли конфігурації модуля iconv для швидкого завантаження.
ldconfig	Налаштовує динамічні зв'язки під час виконання.
ldd	Повідомляє, які спільні бібліотеки потрібні для кожної програми або спільної бібліотеки.
lddlibc4	Допомагає ldd з об'єктними файлами. Не існує на новіших архітектурах, таких як x86_64.
locale	Виводить різну інформацію про поточну локаль
localedef	Компілює специфікації локалі
makedb	Створює просту базу даних з текстового вводу
mtrace	Читає та інтерпретує файл трасування пам'яті та відображає підсумок у форматі, зрозумілому для людини
pcoprofiledump	Виводить інформацію, згенеровану профілюванням ПК
pldd	Перелічує динамічні спільні об'єкти, що використовуються запущеними процесами
sln	Статично пов'язана програма In
sotruss	Відстежує виклики процедур спільніх бібліотек заданої команди
sprof	Читає та відображає дані профілювання спільніх об'єктів
tzselect	Запитує користувача про місцезнаходження системи та повідомляє відповідний часовий пояс опис
xtrace	Відстежує виконання програми, виводячи на екран поточну виконувану функцію
zdump	Думпер часового поясу
zic	Компілятор часового поясу
ld-* .so	Допоміжна програма для виконуваних файлів спільніх бібліотек
libBrokenLocale	Використовується внутрішньо Glibc як грубий хак для запуску пошкоджених програм (наприклад, деяких додатків Motif). Дивіться коментарі в glibc-2.41/locale/broken_cur_max.c для отримання додаткової інформації
libanl	Бібліотека-заглушка, що не містить функцій. Раніше була бібліотекою асинхронного пошуку імен, функції якої тепер знаходяться в libc
libc	Основна бібліотека С

libc_malloc_debug	Увімкнення перевірки розподілу пам'яті при попередньому завантаженні
libdl	Бібліотека-заглушка, що не містить функцій. Раніше була бібліотекою динамічного зв'язування інтерфейсу, функції якої тепер знаходяться в libc
libg	Бібліотека-заглушка, що не містить функцій. Раніше була бібліотекою виконання для g++
libm	Математична бібліотека
libmvec	Бібліотека векторної математики, що підключається за потреби при використанні libm
libmcheck	Вмикає перевірку пам'яті при компонуванні.
libmemusage	Використовується memusage для збору інформації про використання пам'яті програмою
libnsl	Бібліотека мережевих служб, яка зараз є застарілою
libnss_*	Модулі Name Service Switch, що містять функції для вирішення імен хостів, імен користувачів імен груп, псевдонімів, служб, протоколів тощо Завантажується libc відповідно до конфігурації в /etc/nsswitch.conf
libpcprofile	Може бути попередньо завантажена в профіль ПК виконуваного файлу
libpthread	Бібліотека-заглушка, що не містить функцій. Раніше містила функції, що забезпечували більшість інтерфейсів, визначених POSIX.1c Threads Extensions, та інтерфейси семафорів, визначені POSIX.1b Real-time Extensions, зараз ці функції знаходяться в libc
libresolv	Містить функції для створення, надсилання та інтерпретації пакетів до серверів доменних імен Інтернету.
librt	Містить функції, що забезпечують більшість інтерфейсів, визначених POSIX.1b Real-time Extensions.
libthread_db	Містить функції, корисні для створення відладчиків для багатопотокових програм.
libutil	Бібліотека-заглушка, що не містить функцій. Раніше містила код для «стандартних» функцій, що використовуються в багатьох різних утилітах Unix. Зараз ці функції знаходяться в libc

8.6. Zlib-1.3.1

Пакет Zlib містить процедури стиснення та розпакування, які використовуються деякими програмами.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 6.4 MB

8.6.1. Інсталяція Zlib

Підготуйте Zlib до компіляції:

```
./configure --prefix=/usr
```

Зкомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

Видалити непотрібну статичну бібліотеку:

```
rm -fv /usr/lib/libz.a
```

8.6.2. Вміст Zlib

Встановлені бібліотеки: libz.so

Короткі описи

libz

Містить функції стиснення та розпакування, які використовуються деякими програмами.

8.7. Bzip2-1.0.8

Пакет Bzip2 містить програми для стиснення та розпакування файлів. Стиснення текстових файлів за допомогою **bzip2** дає набагато кращий відсоток стиснення, ніж традиційний **gzip**.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 7.2 MB

8.7.1. Інсталяція Bzip2

Застосуйте патч, який встановить документацію для цього пакета:

```
patch -Np1 -i ./bzip2-1.0.8-install_docs-1.patch
```

Наступна команда забезпечує встановлення відносних символічних посилань:

```
sed -i 's@\(\ln -s -f \)${PREFIX}/bin/\!1@' Makefile
```

Переконайтесь, що сторінки та встановлені в правильному місці:

```
sed -i "s@${PREFIX}/man@${PREFIX}/share/man@g" Makefile
```

Підготуйте Bzip2 до компіляції:

```
make -f Makefile-libbz2_so
make clean
```

Значення параметрів make:

-f Makefile-libbz2_so

Це приведе до компіляції Bzip2 з використанням іншого файлу Makefile, в даному випадку файлу Makefile-libbz2_so, який створює динамічну бібліотеку libbz2.so і пов'язує з нею утиліти Bzip2.

Зкомпілюйте пакет:

```
make
```

Встановіть програми:

```
make PREFIX=/usr install
```

Встановіть динамічні бібліотеки:

```
cp -av libbz2.so.* /usr/lib
ln -sv libbz2.so.1.0.8 /usr/lib/libbz2.so
```

Встановіть спільний бінарний файл **bzip2** у каталог /usr/bin і замініть дві копії **bzip2** символічними посиланнями:

```
cp -v bzip2-shared /usr/bin/bzip2
for i in /usr/bin/{bzcat,bunzip2}; do
    ln -sfv bzip2 $i
done
```

Видалити непотрібну статичну бібліотеку:

```
rm -fv /usr/lib/libbz2.a
```

8.7.2. Вміст Bzip2

Встановлені програми:	bunzip2 (посилання на bzip2), bzcat (посилання на bzip2), bzcmp (посилання на bzdiff), bzdif, bzegrep (посилання на bzgrep), bzfgrep (посилання на bzgrep), bzgrep, bzip2, bzip2recover, bzless (посилання на bzmore), та bzmore
Встановлені бібліотеки:	libbz2.so
Встановлені директорії:	/usr/share/doc/bzip2-1.0.8

Короткі описи

bunzip2	Розпаковує файли у форматі bzip
bzcat	Розпаковує у стандартний вивід
bzcmp	Виконує cmp на bzipped файлах
bzdiff	Виконує diff на bzipped файлах
bzegrep	Виконує egrep на bzipped файлах
bzfgrep	Виконує fgrep на bzipped файлах
bzgrep	Виконує grep на bzipped файлах
bzip2	Стискає файли за допомогою алгоритму стиснення тексту з сортуванням блоків Берроуза-Уілера з кодуванням Хаффмана; ступінь стиснення кращий, ніж той, що досягається більш традиційними компресорами, які використовують алгоритми «Лемпеля-Зіва», такі як gzip
bzip2recover	Намагається відновити дані з пошкоджених bzip-файлів
bzless	Запускає less на bzip-файлах
bzmore	Запускає more на файлах, стиснутих за допомогою bzip2.
libbz2	Бібліотека, що реалізує безвтратне стиснення даних із сортуванням блоків за допомогою алгоритму Берроуза-Вілера

8.8. Xz-5.6.4

Пакет Xz містить програми для стиснення та розпакування файлів. Він надає можливості для форматів стиснення lzma та новішого xz. Стиснення текстових файлів за допомогою **xz** забезпечує кращий відсоток стиснення, ніж за допомогою традиційних команд **gzip** або **bzip2**.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 21 MB

8.8.1. Інсталяція Xz

Підготуйте Xz до компіляції:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.6.4
```

Зкомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.8.2. Вміст Xz

Встановлені програми:

lzcat (посилання на xz), lzcmp (посилання на xzdiff), lzdiff (посилання на xzdiff), lzegrep (посилання на xzgrep), lzfgrep (посилання на xzgrep), lzgrep (посилання на xzgrep), lzless (посилання на xzless), lzma (посилання на xz), lzmadec, lzmainfo, lzmore (посилання на xzmore), unlzma (посилання на xz), unxz (link to xz), xz, xzcat (посилання на xz), xzcmp (посилання на xzdiff), xzdec, xzdiff, xzegrep (посилання на xzgrep), xzfgrep (посилання на xzgrep), xzgrep, xzless, and xzmore

Встановлені бібліотеки:

liblzma.so

Встановлені директорії:

/usr/include/lzma та /usr/share/doc/xz-5.6.4

Короткі описи

lzcat	Розпаковує у стандартний вивід.
lzcmp	Запускає cmp на файлах, стиснутих за допомогою LZMA.
lzdiff	Виконує diff на файлах, стиснутих за допомогою LZMA.
lzegrep	Виконує egrep на файлах, стиснутих за допомогою LZMA.
lzfgrep	Виконує fgrep на файлах, стиснутих за допомогою LZMA.

lzgrep	Виконує grep на файлах, стиснутих за допомогою LZMA.
lzless	Запускає less на файлах, стиснутих за допомогою LZMA.
lzma	Стискає або розпаковує файли за допомогою формату LZMA.
lzmadec	Невеликий і швидкий декодер для файлів, стиснутих за допомогою LZMA.
lzmainfo	Показує інформацію, що зберігається в заголовку файлу, стиснутого за допомогою LZMA.
lzmore	Запускає more на файлах, стиснутих за допомогою LZMA.
unlzma	Розпаковує файли у форматі LZMA.
unxz	Розпаковує файли у форматі XZ.
xz	Стискає або розпаковує файли у форматі XZ.
xzcat	Розпаковує у стандартний вивід.
xzcmp	Запускає cmp на файлах, стиснутих за допомогою XZ.
xzdec	Невеликий і швидкий декодер для файлів, стиснутих за допомогою XZ.
xzdiff	Виконує diff на файлах, стиснутих за допомогою XZ.
xzegrep	Виконує egrep на файлах, стиснутих за допомогою XZ.
xzfgrep	Виконує fgrep на файлах, стиснутих за допомогою XZ.
xzgrep	Виконує grep на файлах, стиснутих за допомогою XZ.
xzless	Запускає less на файлах, стиснутих за допомогою XZ.
xzmore	Запускає more на файлах, стиснутих за допомогою XZ.
liblzma	Бібліотека, що реалізує безвтратне стиснення даних із сортуванням блоків за допомогою алгоритму Лемпеля-Зіва-Маркова

8.9. Lz4-1.10.0

Lz4 — це алгоритм стиснення без втрат, що забезпечує швидкість стиснення понад 500 МБ/с на ядро. Він має надзвичайно швидкий декодер зі швидкістю в декілька ГБ/с на ядро. Lz4 може працювати з Zstandard, що дозволяє обом алгоритмам стискати дані швидше.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 4.2 MB

8.9.1. Інсталяція Lz4

Зкомпілюйте пакет:

```
make BUILD_STATIC=no PREFIX=/usr
```

Щоб перевірити результати, виконайте:

```
make -j1 check
```

Встановіть пакет:

```
make BUILD_STATIC=no PREFIX=/usr install
```

8.9.2. Вміст Lz4

Встановлені програми: lz4, lz4c (посилання на lz4), lz4cat (посилання на lz4), and unlz4 (посилання на lz4)

Встановлені бібліотеки: liblz4.so

Короткі описи

lz4 Стискає або розпаковує файли у форматі LZ4.

lz4c Стискає файли у форматі LZ4.

lz4cat Виводить вміст файлу, стиснутого у форматі LZ4.

unlz4 Розпаковує файли у форматі LZ4.

liblz4 Бібліотека, що реалізує стиснення даних без втрат за допомогою алгоритму LZ4.

8.10. Zstd-1.5.7

Zstandard — це алгоритм стиснення в режимі реального часу, що забезпечує високий ступінь стиснення. Він пропонує дуже широкий діапазон компромісів між стисненням і швидкістю, а також підтримується дуже швидким декодером.

Приблизний час побудови: 0.4 SBU

Необхідний простір на диску: 85 MB

8.10.1. Інсталяція Zstd

Зкомпілюйте пакет:

```
make prefix=/usr
```



Note

У результататах тестування є кілька місць, де вказано «failed» (не вдалося). Це очікувані результати, і тільки «FAIL» (НЕВДАЧА) є фактичною невдачею тестування. Не повинно бути жодних невдач тестування.

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make prefix=/usr install
```

Видалити статичну бібліотеку:

```
rm -v /usr/lib/libzstd.a
```

8.10.2. Вміст Zstd

Встановлені програми: zstd, zstdcat (посилання на zstd), zstdgrep, zstdless, zstdmt (посилання на zstd), and unzstd (посилання на zstd)

Встановлені бібліотеки: libzstd.so

Короткі описи

zstd	Стискає або розпаковує файли у форматі ZSTD.
zstdgrep	Запускає grep на стиснутих файлах ZSTD.
zstdless	Запускає less на стиснутих файлах ZSTD.
libzstd	Бібліотека, що реалізує стиснення даних без втрат, використовуючи алгоритм ZSTD.

8.11. File-5.46

Пакет File містить утиліту для визначення типу певного файлу або файлів.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 19 MB

8.11.1. Інсталяція File

Підготуйте File до компіляції:

```
./configure --prefix=/usr
```

Зкомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.11.2. Вміст File

Встановлені програми: file

Встановлені бібліотеки: libmagic.so

Короткі описи

file

Намагається класифікувати кожен наданий файл; для цього виконує кілька тестів — тести файлової системи, тести “магічних чисел” та тести МОВ.

libmagic

Містить процедури для розпізнавання “магічних чисел”, які використовуються програмою **file**.

8.12. Readline-8.2.13

Пакет Readline — це набір бібліотек, що надають можливості редагування командного рядка та історії.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 16 MB

8.12.1. Інсталяція Readline

Перевстановлення Readline призведе до переміщення старих бібліотек до <libraryname>.old. Хоча зазвичай це не є проблемою, в деяких випадках це може спричинити помилку зв'язування в **ldconfig**. Цього можна уникнути, виконавши наступні дві команди sed:

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

Запобігайте жорсткому кодуванню шляхів пошуку бібліотек (rpath) у динамічних бібліотеках. Цей пакет не потребує rpath для встановлення у стандартне місце, а rpath іноді може спричинити небажані ефекти або навіть проблеми з безпекою:

```
sed -i 's/-Wl,-rpath,[^ ]*/' support/shobj-conf
```

Підготуйте Readline до компіляції:

```
./configure --prefix=/usr \
            --disable-static \
            --with-curses \
            --docdir=/usr/share/doc/readline-8.2.13
```

Значення нових конфігураційних параметрів:

--with-curses

Ця опція повідомляє Readline, що функції бібліотеки termcap можна знайти в бібліотеці curses, а не в окремій бібліотеці termcap. Це дозволить створити правильний файл readline.pc.

Зкомпілюйте пакет:

```
make SHLIB_LIBS="-lncursesw"
```

Значення параметрів make:

SHLIB_LIBS=«-lncursesw»

Ця опція змушує Readline зв'язуватися з бібліотекою libncursesw. Детальніше див. розділ «Shared Libraries» у файлі README пакета.

Цей пакет не містить набору тестів.

Встановіть пакет:

```
make install
```

За бажанням встановіть документацію:

```
install -v -m644 doc/*.ps,*.pdf,*.html,*.dvi /usr/share/doc/readline-8.2.13
```

8.12.2. Вміст Readline

Встановлені бібліотеки: libhistory.so and libreadline.so

Встановлені директорії: /usr/include/readline and /usr/share/doc/readline-8.2.13

Короткі описи

libhistory	Надає єдиний інтерфейс користувача для виклику рядків історії
libreadline	Надає набір команд для обробки тексту, введеного в інтерактивній сесії програми

8.13. M4-1.4.19

Пакет M4 містить макропроцесор.

Приблизний час побудови: 0.3 SBU
Необхідний простір на диску: 49 MB

8.13.1. Інсталяція M4

Підготуйте M4 до компіляції:

```
./configure --prefix=/usr
```

Зкомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.13.2. Вміст M4

Встановлені програми: m4

Короткі описи

m4

Копіює вказані файли, розгортаючи макроси, які вони містять. Ці макроси можуть бути вбудованими або визначеними користувачем і можуть приймати будь-яку кількість аргументів. Okрім розгортання макросів, **m4** має вбудовані функції для включення іменованих файлів, виконання команд Unix, виконання арифметичних операцій з цілими числами, маніпулювання текстом, рекурсії тощо. Програма **m4** може використовуватися як інтерфейс для компілятора або як самостійний макропроцесор.

8.14. Bc-7.0.3

Пакет Bc містить мову обробки чисел довільної точності.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 7.8 MB

8.14.1. Інсталяція Bc

Підготуйте Bc до компіляції:

```
CC=gcc ./configure --prefix=/usr -G -O3 -r
```

Значення конфігураційних параметрів:

CC=gcc

Цей параметр визначає компілятор, який буде використовуватися.

-G

Пропускає частини набору тестів, які не працюватимуть, доки не буде встановлено програму bc.

-O3

Вказує оптимізацію, яка буде використовуватися.

-r

Увімкнення використання Readline для поліпшення функції редагування рядків у bc.

Зкомпілюйте пакет:

```
make
```

Щоб перевірити bc, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.14.2. Вміст Bc

Встановлені програми: bc та dc

Короткі описи

bc

Калькулятор командного рядка

dc

Калькулятор командного рядка з оберненою польською нотацією

8.15. Flex-2.6.4

Пакет Flex містить утиліту для створення програм, що розпізнають шаблони в тексті.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 33 MB

8.15.1. Інсталяція Flex

Підготуйте Flex до компіляції:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/flex-2.6.4 \
            --disable-static
```

Зкомпілюйте пакет:

```
make
```

Для перевірки результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

Деякі програми ще не знають про **flex** і намагаються запустити його попередника, **lex**. Щоб допомогти цим програмам, створіть символічне посилання з назвою **lex**, яке запускає **flex** в режимі емуляції **lex**, а також створіть сторінку **man** для **lex** як символічне посилання:

```
ln -sv flex    /usr/bin/lex
ln -sv flex.1  /usr/share/man/man1/lex.1
```

8.15.2. Вміст Flex

Встановлені програми: flex, flex++ (посилання на flex), та lex (посилання на flex)

Встановлені бібліотеки: libfl.so

Встановлені директорії: /usr/share/doc/flex-2.6.4

Короткі описи

flex Інструмент для генерації програм, що розпізнають шаблони в тексті; він дозволяє гнучко задавати правила для пошуку шаблонів, усуваючи необхідність розробки спеціалізованої програми

flex++ Розширення flex, використовується для генерації коду та класів C++. Це символічне посилання на **flex**

lex Символічне посилання, що запускає **flex** в режимі емуляції **lex**

libfl Бібліотека **flex**

8.16. Tcl-8.6.16

Пакет Tcl містить Tool Command Language, надійну мову сценаріїв загального призначення. Пакет Expect написаний на Tcl (вимовляється «тікл»).

Приблизний час побудови: 3.1 SBU

Необхідний простір на диску: 91 MB

8.16.1. Інсталяція Tcl

Цей пакет і наступні два (Expect і DejaGNU) встановлюються для підтримки виконання наборів тестів для Binutils, GCC та інших пакетів. Встановлення трьох пакетів для тестування може здатися надмірним, але дуже важливо, якщо не необхідно, знати, що найважливіші інструменти працюють належним чином.

Підготуйте Tcl до компіляції:

```
SRCDIR=$(pwd)
cd unix
./configure --prefix=/usr \
             --mandir=/usr/share/man \
             --disable-rpath
```

Значення нових конфігураційних параметрів:

`--disable-rpath`

Цей параметр запобігає жорсткому кодуванню шляхів пошуку бібліотек (rpath) у бінарних виконуваних файлах та спільніх бібліотеках. Цей пакет не потребує rpath для встановлення у стандартне місце, а rpath іноді може спричиняти небажані ефекти або навіть проблеми з безпекою.

Зібрати пакет:

```
make

sed -e "s|\$SRCDIR/unix|/usr/lib|" \
      -e "s|\$SRCDIR|/usr/include|" \
      -i tclConfig.sh

sed -e "s|\$SRCDIR/unix/pkgs/tdbc1.1.10|/usr/lib/tdbc1.1.10|" \
      -e "s|\$SRCDIR/pkgs/tdbc1.1.10/generic|/usr/include|" \
      -e "s|\$SRCDIR/pkgs/tdbc1.1.10/library|/usr/lib/tcl8.6|" \
      -e "s|\$SRCDIR/pkgs/tdbc1.1.10|/usr/include|" \
      -i pkgs/tdbc1.1.10/tdbcConfig.sh

sed -e "s|\$SRCDIR/unix/pkgs/itcl4.3.2|/usr/lib/itcl4.3.2|" \
      -e "s|\$SRCDIR/pkgs/itcl4.3.2/generic|/usr/include|" \
      -e "s|\$SRCDIR/pkgs/itcl4.3.2|/usr/include|" \
      -i pkgs/itcl4.3.2/itclConfig.sh

unset SRCDIR
```

Різні інструкції «sed» після команди «make» видаляють посилання на каталог збірки з файлів конфігурації та замінюють їх каталогом інсталяції. Це не є обов'язковим для решти LFS, але може бути необхідним, якщо пакет, зібраний пізніше, використовує Tcl.

Для перевірки результатів, виконайте:

```
make test
```

Встановіть пакет:

```
make install
```

Зробіть встановлену бібліотеку доступною для запису, щоб пізніше можна було видалити символи налагодження:

```
chmod -v u+w /usr/lib/libtcl8.6.so
```

Встановіть заголовки Tcl. Наступний пакет, Expect, потребує їх.

```
make install-private-headers
```

Тепер створіть необхідне символічне посилання:

```
ln -sfv tclsh8.6 /usr/bin/tclsh
```

Перейменувати сторінку довідки, яка конфліктує зі сторінкою довідки Perl:

```
mv /usr/share/man/man3/{Thread,Tcl_Thread}.3
```

За бажанням, встановіть документацію, виконавши наступні команди:

```
cd ..  
tar -xf ../tcl8.6.16-html.tar.gz --strip-components=1  
mkdir -v -p /usr/share/doc/tcl-8.6.16  
cp -v -r ./html/* /usr/share/doc/tcl-8.6.16
```

8.16.2. Вміст Tcl

Встановлені програми: tclsh (посилання на tclsh8.6) та tclsh8.6

Встановлені бібліотеки: libtcl8.6.so and libtclstub8.6.a

Короткі описи

tclsh8.6 Командний інтерпретатор Tcl

tclsh Посилання на tclsh8.6

libtcl8.6.so Бібліотека Tcl

libtclstub8.6.a Бібліотека Tcl Stub

8.17. Expect-5.45.4

Пакет Expect містить інструменти для автоматизації за допомогою скриптових діалогів інтерактивних додатків, таких як telnet, ftp, passwd, fsck, rlogin та tip. Expect також корисний для тестування цих самих додатків, а також для спрощення всіх видів завдань, які є надзвичайно складними для виконання за допомогою будь-яких інших засобів. Фреймворк DejaGnu написаний на Expect.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 3.9 MB

8.17.1. Інсталляція Expect

Expect потребує PTY для роботи. Перевірте, чи PTY працюють належним чином у середовищі chroot, виконавши простий тест:

```
python3 -c 'from pty import spawn; spawn(["echo", "ok"])'
```

Ця команда повинна видати результат ok. Якщо ж натомість у результаті з'явиться OSError: out of pty devices, то середовище не налаштоване для правильної роботи PTY. Вам потрібно вийти з середовища chroot, прочитати розділ 7.3, «Підготовка віртуальних файлових систем ядра» ще раз і переконатися, що файлова система devpts (та інші віртуальні файлові системи ядра) змонтовані правильно. Потім знову увійдіть в середовище chroot, дотримуючись вказівок розділу 7.4, «Вхід в середовище Chroot». Цю проблему необхідно вирішити перед продовженням, інакше набори тестів, що вимагають Expect (наприклад, набори тестів Bash, Binutils, GCC, GDBM і, звичайно, самого Expect), зазнають катастрофічної невдачі, а також можуть статися інші незначні збої.

Тепер внесіть деякі зміни, щоб дозволити пакет із gcc-14.1 або пізнішою версією:

```
patch -Np1 -i ../expect-5.45.4-gcc14-1.patch
```

Підготуйте Expect до компіляції:

```
./configure --prefix=/usr \
            --with-tcl=/usr/lib \
            --enable-shared \
            --disable-rpath \
            --mandir=/usr/share/man \
            --with-tclinclude=/usr/include
```

Значення нових конфігураційних параметрів:

--with-tcl=/usr/lib

Цей параметр необхідний, щоб вказати configure, де знаходиться скрипт tclConfig.sh.

--with-tclinclude=/usr/include

Це явно вказує Expect, де знайти внутрішні заголовки Tcl.

Зкомпілюйте пакет:

```
make
```

Для перевірки результатів, виконайте:

```
make test
```

Встановіть пакет:

```
make install
ln -svf expect5.45.4/libexpect5.45.4.so /usr/lib
```

8.17.2. Вміст Expect

Встановлені програми: expect

Встановлені бібліотеки: libexpect5.45.4.so

Короткі описи

expect	Спілкується з іншими інтерактивними програмами відповідно до сценарію
libexpect-5.45.4.so	Містить функції, які дозволяють використовувати Expect як розширення Tcl або безпосередньо з C або C++ (без Tcl)

8.18. DejaGNU-1.6.3

Пакет DejaGnu містить фреймворк для запуску наборів тестів на інструментах GNU. Він написаний на мові expect, яка сама використовує Tcl (Tool Command Language).

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 6.9 MB

8.18.1. Інсталяція DejaGNU

Upstream рекомендує компілювати DejaGNU у спеціальному каталозі build:

```
mkdir -v build
cd      build
```

Підготуйте DejaGNU до компіляції:

```
./configure --prefix=/usr
makeinfo --html --no-split -o doc/dejagnu.html ../doc/dejagnu.texi
makeinfo --plaintext          -o doc/dejagnu.txt ../doc/dejagnu.texi
```

Для перевірки результатів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
install -v -dm755 /usr/share/doc/dejagnu-1.6.3
install -v -m644 doc/dejagnu.{html,txt} /usr/share/doc/dejagnu-1.6.3
```

8.18.2. Вміст DejaGNU

Встановлені програми: dejagnu та runtest

Короткі описи

dejagnu

Допоміжний запускач команд DejaGNU

runtest

Скрипт-обгортка, який знаходить відповідну оболонку expect, а потім запускає DejaGNU

8.19. Pkgconf-2.3.0

Пакет pkgconf є наступником pkg-config і містить інструмент для передачі шляху включення та/або шляхів бібліотек до інструментів побудови під час етапів конфігурації та створення пакетів.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 4.7 MB

8.19.1. Інсталяція Pkgconf

Підготуйте Pkgconf до компіляції:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/pkgconf-2.3.0
```

Зкомпілюйте пакет:

```
make
```

Встановіть пакет:

```
make install
```

Щоб зберегти сумісність з оригінальним Pkg-config, створіть два символічні посилання:

```
ln -sv pkgconf /usr/bin/pkg-config
ln -sv pkgconf.1 /usr/share/man/man1/pkg-config.1
```

8.19.2. Вміст Pkgconf

Встановлені програми: pkgconf, pkg-config (посилання на pkgconf) та bomtool

Встановлені бібліотеки: libpkgconf.so

Встановлені директорії: /usr/share/doc/pkgconf-2.3.0

Короткі описи

pkgconf

Повертає метаінформацію для вказаної бібліотеки або пакета

bomtool

Генерує специфікацію програмного забезпечення з файлів

libpkgconf

Містить більшість функціональних можливостей pkgconf, дозволяючи іншим інструментам, таким як IDE та компілятори, використовувати його фреймворки

8.20. Binutils-2.44

Пакет Binutils містить компілятор, асемблер та інші інструменти для обробки об'єктів файлів.

Приблизний час побудови: 1.6 SBU

Необхідний простір на диску: 819 MB

8.20.1. Інсталяція Binutils

Документація Binutils рекомендує компілювати Binutils у спеціальному каталозі build:

```
mkdir -v build
cd      build
```

Підготуйте Binutils до компіляції:

```
./configure --prefix=/usr \
           --sysconfdir=/etc \
           --enable-ld=default \
           --enable-plugins \
           --enable-shared \
           --disable-werror \
           --enable-64-bit-bfd \
           --enable-new-dtags \
           --with-system-zlib \
           --enable-default-hash-style=gnu
```

Значення нових конфігураційних параметрів:

--enable-ld=default

Скомпілюйте оригінальний лінкер bfd та встановіть його як ld (лінкер за замовчуванням) та ld.bfd.

--enable-plugins

Увімкніть підтримку плагінів для лінкера.

--with-system-zlib

Використовуйте встановлену бібліотеку zlib замість компіляції включеної версії.

Скомпілюйте пакет:

```
make tooldir=/usr
```

Значення параметрів make:

tooldir=/usr

Зазвичай tooldir (каталог, в якому в кінцевому підсумку будуть розміщені виконувані файли) встановлюється як \$(exec_prefix)/\$(target_alias). Наприклад, для машин x86_64 це буде /usr/x86_64-pc-linux-gnu. Оскільки це є налаштованою системою, цей каталог для конкретної цілі в /usr не є обов'язковим. \$(exec_prefix)/\$(target_alias) буде використовуватися, якщо система використовується для крос-компіляції (наприклад, компіляція пакета на машині Intel, яка генерує код, що може виконуватися на машинах PowerPC).



Important

Набір тестів для Binutils у цьому розділі вважається критично важливим. Не пропускайте його за жодних обставин.

Перевірте результати:

```
make -k check
```

Перевірте результати:

```
grep '^FAIL:' $(find -name '*.log')
```

Встановіть пакет:

```
make tooldir=/usr install
```

Видаліть непотрібні статичні бібліотеки та інші файли:

```
rm -rfv /usr/lib/lib{bfd,ctf,ctf-nobfd,gprofng,opcodes,sframe}.a \
/usr/share/doc/gprofng/
```

8.20.2. Вміст Binutils

Встановлені програми:	addr2line, ar, as, c++filt, dwp, elfedit, gprof, gprofng, ld, ld.bfd, nm, objcopy, objdump, ranlib, readelf, size, strings, and strip
Встановлені бібліотеки:	libbfd.so, libctf.so, libctf-nobfd.so, libgprofng.so, libopcodes.so, та libsframe.so
Встановлені директорії:	/usr/lib/ldscripts

Короткі описи

addr2line

Перетворює адреси програми в імена файлів і номери рядків; за адресою та назвою виконуваного файла використовує інформацію про налагодження у виконуваному файлі, щоб визначити, який вихідний файл і номер рядка пов'язані з цією адресою

ar

Створює, модифікує та витягує з архівів

as

Асемблер, який збирає вихідні дані gcc в об'єктні файли

c++filt

Використовується компонувником для розшифрування символів C++ і Java та запобігання конфлікту перевантажених функцій.

dwp

Утиліта упаковки DWARF

elfedit

Оновлює заголовки ELF файлів ELF

gprof

Відображає дані профілю графіка викликів

gprofng

Збирає та аналізує дані про продуктивність

ld

Компонувник, який об'єднує декілька об'єктних та архівних файлів в один файл, переміщуючи їхні дані та пов'язуючи посилання на символи

ld.bfd	Жорстке посилання на ld
nm	Перелічує символи, що зустрічаються в заданому об'єктному файлі
objcopy	Перетворює один тип об'єктного файлу в інший
objdump	Відображає інформацію про заданий об'єктний файл, з опціями, що контролюють конкретну інформацію, яку потрібно відобразити; відображена інформація корисна для програмістів,
ranlib	Генерує індекс вмісту архіву і зберігає його в архіві; індекс перелічує всі символи, визначені членами архіву, які є переміщуваними об'єктними файлами
readelf	Відображає інформацію про бінарні файли типу ELF
size	Перелічує розміри секцій і загальний розмір заданих об'єктних файлів
strings	Виводить для кожного заданого файла послідовності друкованих символів, які мають принаймні задану довжину (за замовчуванням чотири); для об'єктних файлів за замовчуванням виводить тільки рядки з розділів ініціалізації та завантаження, тоді як для інших типів файлів сканує весь файл
strip	Відкидає символи з об'єктних файлів
libbfd	Бібліотека опису бінарних файлів
libctf	Бібліотека підтримки налагодження формату типу Compat ANSI-C
libctf-nobfd	Варіант libctf, який не використовує функціональність libbfd
libgprofng	Бібліотека, що містить більшість процедур, які використовує gprofng
libopcodes	Бібліотека для роботи з кодами операцій — «читабельними текстовими» версіями інструкцій для процесора; використовується для створення утиліт, таких як objdump
libsframe	Бібліотека для підтримки онлайн-відстеження за допомогою простого розмотувача

8.21. GMP-6.3.0

Пакет GMP містить математичні бібліотеки. Вони мають корисні функції для арифметики довільної точності.

Приблизний час побудови: 0.3 SBU

Необхідний простір на диску: 54 MB

8.21.1. Інсталяція GMP-6.3.0



Note

Якщо ви створюєте 32-бітну x86-версію, але маєте процесор, здатний виконувати 64-бітний код, і ви вказали `CFLAGS` у середовищі, скрипт `configure` спробує налаштувати 64-бітну версію і зазнає невдачі. Уникнути цього можна, запустивши команду `configure` нижче з

```
ABI=32 ./configure ...
```



Note

За замовчуванням GMP створює бібліотеки, оптимізовані для процесора хоста. Якщо потрібні бібліотеки, придатні для процесорів, менш потужних за процесор хоста, можна створити загальні бібліотеки, додавши опцію `--host=none-linux-gnu` до команди `configure`.

Підготуйте GMP-6.3.0 до компіляції:

```
./configure --prefix=/usr \
            --enable-cxx \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.3.0
```

Значення нових конфігураційних параметрів:

`--enable-cxx`

Цей параметр увімкне підтримку C++

`--docdir=/usr/share/doc/gmp-6.3.0`

Ця змінна вказує правильне місце для документації.

Скомпілюйте пакет і створіть документацію у форматі HTML:

```
make
make html
```



Important

Набір тестів для GMP у цьому розділі вважається критично важливим. Не пропускайте його за жодних обставин.

Перевірте результати:

```
make check 2>&1 | tee gmp-check-log
```



Caution

Код у gmp є високо оптимізованим для процесора, на якому він побудований. Іноді код, що визначає процесор, неправильно ідентифікує можливості системи, і в тестах або інших програмах, що використовують бібліотеки gmp, виникають помилки з повідомленням `Illegal instruction`. У цьому випадку gmp слід переконфігурувати з опцією `--host=none-linux-gnu` і перекомпілювати.

Переконайтесь, що принаймні 199 тестів у наборі тестів пройшли успішно. Перевірте результати, виконавши таку команду:

```
awk '/# PASS:/ {total+=$3} ; END{print total}' gmp-check-log
```

Встановіть пакет та його документацію:

```
make install  
make install-html
```

8.21.2. Вміст GMP-6.3.0

Встановлені бібліотеки: libgmp.so та libgmpxx.so

Встановлені директорії: /usr/share/doc/gmp-6.3.0

Короткі описи

libgmp

Містить функції точної математики

libgmpxx

Містить функції точної математики C++

8.22. MPFR-4.2.1

Пакет MPFR містить функції для математичних обчислень з довільною точністю.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 43 MB

8.22.1. Інсталяція MPFR

Підготуйте MPFR до компіляції:

```
./configure --prefix=/usr \
            --disable-static \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.2.1
```

Скомпілюйте пакет і створіть документацію у форматі HTML:

```
make
make html
```



Important

Набір тестів для MPFR у цьому розділі вважається критично важливим. Не пропускайте його за жодних обставин.

Перевірте результати та переконайтесь, що всі 198 тестів пройшли успішно:

```
make check
```

Встановіть пакет та його документацію:

```
make install
make install-html
```

8.22.2. Вміст MPFR

Встановлені бібліотеки: libmpfr.so

Встановлені директорії: /usr/share/doc/mpfr-4.2.1

Короткі описи

libmpfr

Містить математичні функції з довільною точністю.

8.23. MPC-1.3.1

Пакет MPC містить бібліотеку для арифметики комплексних чисел з довільно високою точністю і правильним округленням результату.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 22 MB

8.23.1. Інсталяція MPC

Підготуйте MPC до компіляції:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.3.1
```

Скомпілюйте пакет і створіть документацію у форматі HTML:

```
make
make html
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет та його документацію:

```
make install
make install-html
```

8.23.2. Вміст MPC

Встановлені бібліотеки: libmpc.so

Встановлені директорії: /usr/share/doc/mpc-1.3.1

Короткі описи

libmpc

Містить складні математичні функції

8.24. Attr-2.5.2

Пакет Attr містить утиліти для адміністрування розширених атрибутів об'єктів файлової системи.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 4.1 MB

8.24.1. Інсталяція Attr

Підготуйте Attr до компіляції:

```
./configure --prefix=/usr \
            --disable-static \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.5.2
```

Скомпілюйте пакет:

```
make
```

Тести повинні виконуватися на файловій системі, яка підтримує розширені атрибути, такі як файлові системи ext2, ext3 або ext4. Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.24.2. Вміст Attr

Встановлені програми: attr, getfattr та setfattr

Встановлені бібліотеки: libattr.so

Встановлені директорії: /usr/include/attr та /usr/share/doc/attr-2.5.2

Короткі описи

attr Розширює атрибути об'єктів файлової системи

getfattr Отримує розширені атрибути об'єктів файлової системи

setfattr Встановлює розширені атрибути об'єктів файлової системи

libattrc Містить бібліотечні функції для роботи з розширеними атрибутами

8.25. Acl-2.3.2

Пакет Acl містить утиліти для адміністрування списків контролю доступу, які використовуються для визначення детальних дискреційних прав доступу до файлів і каталогів.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 6.5 MB

8.25.1. Інсталяція Acl

Підготуйте Acl до компіляції:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/acl-2.3.2
```

Скомпілюйте пакет:

```
make
```

Тести Acl повинні виконуватися на файловій системі, яка підтримує контроль доступу. Щоб перевірити результати, виконайте:

```
make check
```

Відомо, що один тест під назвою `test/cp.test` не проходить, оскільки Coreutils ще не побудовано з підтримкою Acl.

Встановіть пакет:

```
make install
```

8.25.2. Вміст Acl

Встановлені програми: chacl, getfacl, та setfacl

Встановлені бібліотеки: libacl.so

Встановлені директорії: /usr/include/acl та /usr/share/doc/acl-2.3.2

Короткі описи

chacl	Змінює список контролю доступу до файлу або каталогу
getfacl	Отримує списки контролю доступу до файлів
setfacl	Встановлює списки контролю доступу до файлів
libacl	Містить бібліотечні функції для роботи зі списками контролю доступу

8.26. Libcap-2.73

Пакет Libcap реалізує інтерфейс користувачького простору для можливостей POSIX 1003.1e, доступних у ядрах Linux. Ці можливості розділяють всесильні привілеї root на встановлення окремих привілеїв.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 3.0 MB

8.26.1. Інсталяція Libcap

Запобігання встановленню статичних бібліотек:

```
sed -i '/install -m.*STA/d' libcap/Makefile
```

Скомпілюйте пакет:

```
make prefix=/usr lib=lib
```

Значення параметрів make:

lib=lib

Цей параметр встановлює каталог бібліотеки на */usr/lib* замість */usr/lib64* на x86_64. Він не має впливу на x86.

Щоб перевірити результати, виконайте:

```
make test
```

Встановіть пакет:

```
make prefix=/usr lib=lib install
```

8.26.2. Вміст Libcap

Встановлені програми: capsh, getcap, getpcaps та setcap

Встановлені бібліотеки: libcap.so та libpsx.so

Короткі описи

capsh	Оболонка для дослідження та обмеження підтримки можливостей
getcap	Перевіряє можливості файлів
getpcaps	Відображає можливості запитуваних процесів
setcap	Встановлює можливості файлів
libcap	Містить бібліотечні функції для маніпулювання можливостями POSIX 1003.1e
libpsx	Містить функції для підтримки семантики POSIX для системних викликів, пов'язаних з бібліотекою pthread

8.27. Libxcrypt-4.4.38

Пакет Libxcrypt містить сучасну бібліотеку для одностороннього хешування паролів.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 12 MB

8.27.1. Інсталяція Libxcrypt

Підготуйте Libxcrypt до компіляції:

```
./configure --prefix=/usr \
            --enable-hashes=strong,glibc \
            --enable-obsolete-api=no \
            --disable-static \
            --disable-failure-tokens
```

Значення нових конфігураційних параметрів:

--enable-hashes=strong, glibc

Створює сильні алгоритми хешування, рекомендовані для випадків використання з метою безпеки, та алгоритми хешування, що надаються традиційною Glibc libcrypt для сумісності.

--enable-obsolete-api=no

Вимкнути застарілі функції API. Вони не потрібні для сучасної системи Linux, побудованої з вихідного коду.

--disable-failure-tokens

Вимкнути функцію токенів помилок. Вона потрібна для сумісності з традиційними бібліотеками хешування деяких платформ, але система Linux на базі Glibc її не потребує.

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```



Note

Вищезазначені інструкції вимкнули застарілі функції API, оскільки жоден пакет, встановлений шляхом компіляції з вихідних кодів, не буде пов'язаний з ними під час виконання. Однак єдині відомі бінарні програми, які пов'язані з цими функціями, вимагають ABI версії 1. Якщо вам необхідні такі функції через наявність бінарної програми або для відповідності LSB, перекомпілюйте пакет за допомогою наступних команд:

```
make distclean
./configure --prefix=/usr \
             --enable-hashes=strong,glibc \
             --enable-obsolete-api=glibc \
             --disable-static \
             --disable-failure-tokens
make
cp -av --remove-destination .libs/libcrypt.so.1* /usr/lib
```

8.27.2. Вміст Libxcrypt

Встановлені бібліотеки: libcrypt.so

Короткі описи

libcrypt	Містить функції хешування паролів
----------	-----------------------------------

8.28. Shadow-4.17.3

Пакет Shadow містить програми для безпечної обробки паролів.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 114 MB

8.28.1. Інсталяція Shadow



Important

Якщо ви встановили Linux-PAM, вам слід дотримуватися інструкцій BLFS замість цієї сторінки, щоб скомпілювати (або перекомпілювати чи оновити) shadow.



Note

Якщо ви хочете забезпечити використання надійних паролів, спочатку *встановіть і налаштуйте Linux-PAM*. Потім *встановіть і налаштуйте shadow з підтримкою PAM*. Нарешті, *встановіть libpwquality і налаштуйте PAM для його використання*.

Вимкніть встановлення програми **groups** та її сторінок довідки, оскільки Coreutils надає кращу версію. Також запобігайте встановленню сторінок довідки, які вже були встановлені в розділі 8.3, «Man-pages-6.12»:

```
sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i 's/groups\.\.1 / /' {} \;
find man -name Makefile.in -exec sed -i 's/getspnam\.\.3 / /' {} \;
find man -name Makefile.in -exec sed -i 's/passwd\.\.5 / /' {} \;
```

Замість використання стандартного методу *crypt*, використовуйте набагато більш безпечний метод шифрування паролів *YESCRYPT*, який також дозволяє використовувати паролі довжиною більше 8 символів. Також необхідно змінити застаріле розташування */var/spool/mail* для поштових скриньок користувачів, яке Shadow використовує за замовчуванням, на розташування */var/mail*, яке використовується в даний час. Крім того, видаліть */bin* і */sbin* з PATH, оскільки вони є просто символічними посиланнями на їхні відповідники в */usr*.



Warning

Включення */bin* та/або */sbin* до змінної PATH може привести до того, що деякі пакети BLFS не зможуть скомпілюватися, тому не робіть цього у файлі *.bashrc* або деінде.

```
sed -e 's:#ENCRYPT_METHOD DES:ENCRYPT_METHOD YESCRYPT:' \
-e 's:/var/spool/mail:/var/mail:' \
-e '/PATH=/ {s@/sbin:@@;s@/bin:@@}' \
-i etc/login.defs
```

Підготуйте Shadow до компіляції:

```
touch /usr/bin/passwd
./configure --sysconfdir=/etc \
--disable-static \
--with-{b,yes}crypt \
--without-libbsd \
--with-group-name-max-length=32
```

Значення нових конфігураційних параметрів:

touch /usr/bin/passwd

Файл `/usr/bin/passwd` повинен існувати, оскільки його розташування жорстко задано в деяких програмах; якщо він ще не існує, інсталяційний скрипт створить його в неправильному місці.

--with-{b,yes}crypt

Оболонка розширює це до двох перемикачів, `--with-bcrypt` та `--with-yescrypt`. Вони дозволяють shadow використовувати алгоритми Bcrypt та Yescrypt, реалізовані Libxcrypt для хешування паролів. Ці алгоритми є більш безпечними (зокрема, набагато більш стійкими до атак на основі GPU) ніж традиційні алгоритми SHA.

--with-group-name-max-length=32

Найдовше допустиме ім'я користувача — 32 символи. Встановити таку ж максимальну довжину для імені групи.

--without-libbsd

Не використовувати функцію `readpassphrase` з `libbsd`, якої немає в LFS. Замість цього використовувати внутрішню копію.

Скомпілюйте пакет:

```
make
```

Цей пакет не містить набору тестів.

Встановіть пакет:

```
make exec_prefix=/usr install
make -C man install-man
```

8.28.2. Конфігурація Shadow

Цей пакет містить утиліти для додавання, модифікації та видалення користувачів і груп; встановлення та зміни їхніх паролів; а також виконання інших адміністративних завдань. Повне пояснення того, що означає «затінення паролів», див. у файлі `doc/HOWTO` в розпакованому дереві джерел. Якщо ви використовуєте підтримку Shadow, майте на увазі, що програми, які потребують перевірки паролів (менеджери відображення, програми FTP, демони pop3 тощо), повинні бути сумісними з Shadow. Тобто вони повинні вміти працювати з затіненими паролями.

Щоб увімкнути тіньові паролі, виконайте таку команду:

```
pwconv
```

Щоб увімкнути тіньові паролі груп, виконайте:

```
grpconv
```

Стандартна конфігурація Shadow для утиліти **useradd** потребує деяких пояснень. По-перше, стандартною дією утиліти **useradd** є створення користувача та групи з таким самим іменем, як і користувач. За замовчуванням ідентифікатор користувача (UID) та ідентифікатор групи (GID) починаються з 1000. Це означає, що якщо ви не передаєте додаткові параметри до **useradd**, кожен користувач буде членом унікальної групи в системі. Якщо така поведінка є небажаною, вам потрібно передати параметр **-g** або **-n** до **useradd**, або змінити налаштування **USERGROUPS_ENAB** у **/etc/login.defs**. Дивіться **useradd(8)** для додаткової інформації.

По-друге, щоб змінити параметри за замовчуванням, потрібно створити файл **/etc/default/useradd** і налаштовувати його відповідно до ваших потреб. Створіть його за допомогою:

```
mkdir -p /etc/default
useradd -D --gid 999
```

Пояснення параметрів **/etc/default/useradd**:

GROUP=999

Цей параметр встановлює початок номерів груп, що використовуються у файлі **/etc/group**. Конкретне значення 999 походить від параметра **--gid** вище. Ви можете встановити будь-яке бажане значення. Зверніть увагу, що **useradd** ніколи не буде повторно використовувати UID або GID. Якщо номер, визначений у цьому параметрі, вже використовується, буде використано наступний доступний номер. Зверніть також увагу, що якщо у вашій системі немає групи з ідентифікатором, рівним цьому номеру, то при першому використанні **useradd** без параметра **-g** буде згенеровано повідомлення про помилку — **useradd: unknow GID 999**, навіть якщо обліковий запис було створено правильно. Саме тому ми створили групу **users** з цим ідентифікатором групи в розділі 7.6, «Створення необхідних файлів і символічних посилань».

CREATE_MAIL_SPOOL=yes

Цей параметр змушує **useradd** створювати файл поштової скриньки для кожного нового користувача. **useradd** призначить групову власність цього файлу групі **mail** з правами доступу 0660. Якщо ви не хочете створювати ці файли, виконайте наступну команду:

```
sed -i '/MAIL/s/yes/no/' /etc/default/useradd
```

8.28.3. Налаштування пароля root

Виберіть пароль для користувача **root** і встановіть його, виконавши:

```
passwd root
```

8.28.4. Вміст Shadow

Встановлені програми: chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, getsubids, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, login, logoutd, newgidmap, newgrp, newuidmap, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (посилання на **newgrp**), su, useradd, userdel, usermod, vigr (посилання на **vipw**) та **vipw**

Встановлені бібліотеки: libsubid.so

Встановлені директорії: /etc/default та /usr/include/shadow

Короткі описи

chage	Використовується для зміни максимальної кількості днів між обов'язковими змінами пароля.
chfn	Використовується для зміни повного імені користувача та іншої інформації.
chgpasswd	Використовується для оновлення паролів групи в пакетному режимі.
chpasswd	Використовується для оновлення паролів користувачів в пакетному режимі.
chsh	Використовується для зміни оболонки входу за замовчуванням користувача.
expiry	Перевіряє та застосовує поточну політику закінчення терміну дії пароля.
faillog	Використовується для перевірки журналу невдалих спроб входу, встановлення максимальної кількості невдалих спроб перед блокуванням облікового запису та скидання лічильника невдалих спроб.
getsubids	Використовується для переліку підлеглих діапазонів ідентифікаторів для користувача.
gpasswd	Використовується для додавання та видалення членів і адміністраторів груп
groupadd	Створює групу з заданим іменем
groupdel	Видаляє групу з заданим іменем
groupmems	Дозволяє користувачеві адмініструвати власний список членів групи без необхідності мати права суперкористувача
groupmod	Використовується для зміни імені або GID заданої групи.
grpck	Перевіряє цілісність файлів груп /etc/group та /etc/gshadow.
grpconv	Створює або оновлює файл тіньової групи з нормального файлу групи.
grpunconv	Оновлює /etc/group з /etc/gshadow, а потім видаляє останній.
login	Використовується системою для дозволення входу користувачів
logoutd	Є демоном, що використовується для застосування обмежень часу входу та портів
newgidmap	Використовується для встановлення відповідності gid простору імен користувача
newgrp	Використовується для зміни поточного GID під час сеансу входу
newuidmap	Використовується для встановлення відповідності uid простору імен користувача
newusers	Використовується для створення або оновлення цілої серії облікових записів користувачів

nologin	Відображає повідомлення про те, що обліковий запис недоступний; призначена для використання як оболонка за замовчуванням для вимкнених облікових записів
passwd	Використовується для зміни пароля облікового запису користувача або групи
pwck	Перевіряє цілісність файлів паролів /etc/passwd та /etc/shadow
pwconv	Створює або оновлює файл тіньових паролів із звичайного файлу паролів
pwunconv	Оновлює /etc/passwd із /etc/shadow, а потім видаляє останній
sg	Виконує задану команду, коли GID користувача встановлено на значення заданої групи
su	Запускає оболонку із заміщеними ідентифікаторами користувача та групи
useradd	Створює нового користувача із заданим ім'ям або оновлює інформацію про нового користувача за замовчуванням
userdel	Видаляє вказаний обліковий запис користувача
usermod	Використовується для зміни імені входу, ідентифікатора користувача (UID), оболонки, початкової групи, домашнього каталогу тощо заданого користувача
vigr	Редагує файли /etc/group або /etc/gshadow
vipw	Редагує файли /etc/passwd або /etc/shadow
libsubid	Бібліотека для обробки підлеглих діапазонів ідентифікаторів для користувачів і груп

8.29. GCC-14.2.0

Пакет GCC містить колекцію компіляторів GNU, яка включає компілятори C і C++.

Приблизний час побудови: 46 SBU (разом з тестами)

Необхідний простір на диску: 6.3 GB

8.29.1. Інсталяція GCC

Якщо ви використовуєте x86_64, змініть ім'я каталогу за замовчуванням для 64-бітних бібліотек на «lib»:

```
case $(uname -m) in
x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
;;
esac
```

Документація GCC рекомендує компілювати GCC у окремому каталозі build:

```
mkdir -v build
cd      build
```

Підготуйте GCC до компіляції:

```
./configure --prefix=/usr \
LD=ld \
--enable-languages=c,c++ \
--enable-default-pie \
--enable-default-ssp \
--enable-host-pie \
--disable-multilib \
--disable-bootstrap \
--disable-fixincludes \
--with-system-zlib
```

GCC підтримує сім різних мов програмування, але необхідні компоненти для більшості з них ще не встановлені. Інструкції щодо компіляції всіх мов, що підтримуються GCC, дивіться на сторінці BLFS Book GCC.

Значення нових конфігураційних параметрів:

LD=ld

Цей параметр змушує скрипт конфігурації використовувати програму ld, встановлену пакетом Binutils, зібраним раніше в цьому розділі, а не крос-збірну версію, яка б інакше використовувалася.

--disable-fixincludes

За замовчуванням, під час інсталяції GCC деякі системні заголовки будуть «виправлені» для використання з GCC. Це не є необхідним для сучасної системи Linux і може бути шкідливим, якщо пакет перевстановлюється після встановлення GCC. Цей перемикач запобігає «фіксуванню» заголовків GCC.

--with-system-zlib

Цей перемикач вказує GCC зв'язуватися з встановленою в системі копією бібліотеки Zlib, а не з власною внутрішньою копією.



Note

PIE (позиційно-незалежні виконувані файли) — це бінарні програми, які можна завантажувати в будь-яке місце пам'яті. Без PIE функцію безпеки під назвою ASLR (Address Space Layout Randomization, рандомізація розміщення адресного простору) можна застосовувати для динамічних бібліотек, але не для самих виконуваних файлів. Увімкнення PIE дозволяє застосовувати ASLR для виконуваних файлів на додаток до динамічних бібліотек і пом'якшує деякі атаки, засновані на фікованих адресах чутливого коду або даних у виконуваних файлах.

SSP (Stack Smashing Protection) — це техніка, що забезпечує захист стека параметрів від пошкодження. Пошкодження стека може, наприклад, змінити адресу повернення підпрограми, тим самим передаючи контроль деякому небезпечному коду (існуючому в програмі або динамічних бібліотеках, або введенному зловмисником якимось чином).

Скомпілюйте пакет:

```
make
```



Important

У цьому розділі набір тестів для GCC вважається важливим, але його виконання займає багато часу. Новачкам рекомендується запустити набір тестів. Час виконання тестів можна значно скоротити, додавши `-jx` до команди `make -k check` нижче, де x — кількість ядер процесора у вашій системі.

GCC може потребувати більше місця в стеку для компіляції деяких надзвичайно складних кодових шаблонів. Як запобіжний захід для дистрибутивів хоста з жорстким обмеженням стеку, явно встановіть жорстке обмеження розміру стеку на нескінченність. У більшості дистрибутивів хоста (та кінцевої системи LFS) жорсткий ліміт за замовчуванням є нескінченим, але явне встановлення не завдасть шкоди. Немає необхідності змінювати м'який ліміт розміру стека, оскільки GCC автоматично встановить його на відповідне значення, якщо воно не перевищує жорсткий ліміт:

```
ulimit -s -H unlimited
```

Тепер усуньте/вправте кілька відомих помилок тестування:

```
sed -e '/cpython/d'           -i ..../gcc/testsuite/gcc.dg/plugin/plugin.exp
sed -e 's/no-pic /&-no-pie /' -i ..../gcc/testsuite/gcc.target/i386/pr113689-1.c
sed -e 's/300000/(1|300000)/' -i ..../libgomp/testsuite/libgomp.c-c++-common/pr109062.c
sed -e 's/{ target nonpic } //'\n-e '/GOTPCREL/d'          -i ..../gcc/testsuite/gcc.target/i386/fentryname3.c
```

Перевірте результати як користувач без привілеїв, але не зупиняйтесь на помилках:

```
chown -R tester .
su tester -c "PATH=$PATH make -k check"
```

Щоб отримати підсумок результатів тестового набору, виконайте:

```
../contrib/test_summary
```

Щоб відфільтрувати тільки підсумки, пропустіть вихідні дані через `grep -A7 Summ.`

Результати можна порівняти з тими, що знаходяться за адресами <https://www.linuxfromscratch.org/lfs/build-logs/12.3/> та <https://gcc.gnu.org/ml/gcc-testresults/>.

Відомо, що тести tsan не проходять на деяких дистрибутивах хостів.

Деяких несподіваних невдач не завжди можна уникнути. У деяких випадках невдачі тестів залежать від конкретного обладнання системи.

Якщо результати тестів не сильно відрізняються від тих, що знаходяться за вищевказаними URL-адресами, можна безпечно продовжувати.

Встановіть пакет:

```
make install
```

Каталог збірки GCC тепер належить `tester`, а права власності на встановлений каталог заголовків (та його вміст) є неправильними. Змініть права власності на користувача `root` та групу:

```
chown -v -R root:root \
  /usr/lib/gcc/$(gcc -dumpmachine)/14.2.0/include{,-fixed}
```

Створити символічне посилання, необхідне *FHS* з «історичних» причин.

```
ln -svr /usr/bin/cpp /usr/lib
```

Багато пакетів використовують ім'я `cc` для виклику компілятора С. Ми вже створили `cc` як символічне посилання в `gcc-pass2`, створіть його сторінку `man` також як символічне посилання:

```
ln -sv gcc.1 /usr/share/man/man1/cc.1
```

Додайте символічне посилання сумісності, щоб увімкнути компіляцію програм із оптимізацією часу зв'язування (LTO):

```
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/14.2.0/liblto_plugin.so \
  /usr/lib/bfd-plugins/
```

Тепер, коли наш остаточний набір інструментів готовий, важливо ще раз переконатися, що компіляція та зв'язування працюватимуть як очікується. Для цього ми виконуємо кілька перевірок працездатності:

```
echo 'int main(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

Помилок бути не повинно, і результат виконання останньої команди буде таким (з урахуванням відмінностей у назвах динамічних лінкерів для різних платформ):

```
[Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
```

Тепер переконайтесь, що ми налаштовані на використання правильних файлів запуску:

```
grep -E -o '/usr/lib.*?crt[1lin].*succeeded' dummy.log
```

Результатом виконання останньої команди має бути:

```
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/Scrt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/crtn.o succeeded
```

Залежно від архітектури вашої машини, вищезазначене може дещо відрізнятися. Відмінність буде полягати в назві каталогу після `/usr/lib/gcc`. Важливо звернути увагу на те, що `gcc` знайшов усі три файли `crt*.o` в каталозі `/usr/lib`.

Перевірте, чи компілятор шукає правильні файли заголовків:

```
grep -B4 '^ /usr/include' dummy.log
```

Ця команда повинна повернути наступний результат:

```
#include <...> search starts here:  
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/include  
/usr/local/include  
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/include-fixed  
/usr/include
```

Знову ж таки, каталог, названий на честь вашої цільової трійки, може відрізнятися від наведеного вище, залежно від архітектури вашої системи.

Далі переконайтесь, що новий лінкер використовується з правильними шляхами пошуку:

```
grep 'SEARCH.* /usr/lib' dummy.log | sed 's|;|\\n|g'
```

Посилання на шляхи, що містять компоненти з «`-linux-gnu`», слід ігнорувати, але в іншому випадку вихідні дані останньої команди повинні бути такими:

```
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib64")  
SEARCH_DIR("/usr/local/lib64")  
SEARCH_DIR("/lib64")  
SEARCH_DIR("/usr/lib64")  
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib")  
SEARCH_DIR("/usr/local/lib")  
SEARCH_DIR("/lib")  
SEARCH_DIR("/usr/lib");
```

32-бітна система може використовувати кілька інших каталогів. Наприклад, ось вихідні дані з машини i686:

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib32")  
SEARCH_DIR("/usr/local/lib32")  
SEARCH_DIR("/lib32")  
SEARCH_DIR("/usr/lib32")  
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")  
SEARCH_DIR("/usr/local/lib")  
SEARCH_DIR("/lib")  
SEARCH_DIR("/usr/lib");
```

Далі переконайтесь, що ми використовуємо правильну бібліотеку `libc`:

```
grep "/lib.* /libc.so.6" dummy.log
```

Результатом виконання останньої команди має бути:

```
attempt to open /usr/lib/libc.so.6 succeeded
```

Переконайтесь, що `GCC` використовує правильний динамічний лінкер:

```
grep found dummy.log
```

Результатом виконання останньої команди має бути (з урахуванням відмінностей у назвах динамічних лінкерів для різних платформ):

```
found ld-linux-x86-64.so.2 at /usr/lib/ld-linux-x86-64.so.2
```

Якщо вихідні дані не відображаються, як показано вище, або не отримуються взагалі, то це означає, що щось серйозно не так. Проведіть розслідування та простежте кроки, щоб з'ясувати, де саме полягає проблема, і вправте її. Перед продовженням процесу необхідно вирішити всі проблеми.

Коли все працює правильно, очистіть тестові файли:

```
rm -v dummy.c a.out dummy.log
```

На завершення, перемістіть файл, який знаходиться не на своєму місці:

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

8.29.2. Вміст GCC

Встановлені програми:	c++, cc (посилання на gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, gcov, gcov-dump, gcov-tool, and lto-dump
Встановлені бібліотеки:	libasan.{a,so}, libatomic.{a,so}, libgcc1.so, libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, libhwasan.{a,so}, libitm.{a,so}, liblsan.{a,so}, liblto_plugin.so, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so}, libstdc++exp.a, libstdc++fs.a, libsupc++.a, libtsan.{a,so}, and libubsan.{a,so}
Встановлені директорії:	/usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc та /usr/share/gcc-14.2.0

Короткі описи

c++	Компілятор C++
cc	Компілятор C
cpp	Препроцесор C; використовується компілятором для розширення директив #include, #define та подібних у вихідних файлах
g++	Компілятор C++
gcc	Компілятор C
gcc-ar	Оболонка навколо ar , яка додає плагін до командного рядка. Ця програма використовується тільки для додавання «оптимізації часу зв'язування» і не є корисною з опціями побудови за замовчуванням.
gcc-nm	Оболонка навколо nm , яка додає плагін до командного рядка. Ця програма використовується тільки для додавання «оптимізації часу зв'язку» і не є корисною з опціями побудови за замовчуванням.
gcc-ranlib	Оболонка навколо ranlib , яка додає плагін до командного рядка. Ця програма використовується тільки для додавання «оптимізації часу зв'язку» і не є корисною з опціями побудови за замовчуванням.

gcov	Інструмент тестування покриття; використовується для аналізу програм з метою визначення, де оптимізації матимуть найбільший ефект
gcov-dump	Інструмент для вивантаження профілів gcda та gcpo в автономному режимі
gcov-tool	Інструмент для обробки профілів gcda в автономному режимі
lto-dump	Інструмент для вивантаження об'єктних файлів, створених GCC з увімкненою LTO
libasan	Бібліотека виконання Address Sanitizer
libatomic	Вбудована бібліотека виконання GCC
libcc1	Бібліотека, яка дозволяє GDB використовувати GCC
libgcc	Містить підтримку виконання для gcc
libgcov	Ця бібліотека підключається до програми, коли GCC отримує вказівку увімкнути профілювання
libgomp	GNU-реалізація API OpenMP для багатоплатформного паралельного програмування з спільною пам'яттю в C/C++ та Fortran
libhwasan	Бібліотека виконання Address Sanitizer з апаратним забезпеченням
libitm	Бібліотека транзакційної пам'яті GNU
liblsan	Бібліотека виконання Leak Sanitizer
liblto_plugin	Плагін LTO GCC дозволяє Binutils обробляти об'єктні файли, створені GCC з увімкненим LTO
libquadmath	API бібліотеки математичних функцій з чотирикратною точністю GCC
libssp	Містить процедури, що підтримують функціональність захисту GCC від руйнування стека. Зазвичай вона не використовується, оскільки Glibc також надає ці процедури.
libstdc++	Стандартна бібліотека C++
libstdc++exp	Експериментальна бібліотека контрактів C++
libstdc++fs	ISO/IEC TS 18822:2015 Бібліотека файлової системи
libsupc++	Надає процедури підтримки для мови програмування C++
libtsan	Бібліотека виконання Thread Sanitizer
libubsan	Бібліотека виконання Undefined Behavior Sanitizer

8.30. Ncurses-6.5

Пакунок Ncurses містить бібліотеки для незалежної від терміналу обробки символьних екранів.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 46 MB

8.30.1. Інсталяція Ncurses

Підготуйте Ncurses до компіляції:

```
./configure --prefix=/usr \
            --mandir=/usr/share/man \
            --with-shared \
            --without-debug \
            --without-normal \
            --with-cxx-shared \
            --enable-pc-files \
            --with-pkg-config-libdir=/usr/lib/pkgconfig
```

Значення нових конфігураційних параметрів:

--with-shared

Це змушує Ncurses компілювати та встановлювати спільні бібліотеки C.

--without-normal

Це запобігає компіляції та встановленню Ncurses статичних бібліотек C.

--without-debug

Це запобігає компіляції та встановленню Ncurses бібліотек для налагодження.

--with-cxx-shared

Це змушує Ncurses компілювати та встановлювати спільні зв'язки C++. Це також запобігає компіляції та встановленню статичних зв'язків C++.

--enable-pc-files

Цей перемикач генерує та встановлює файли .pc для pkg-config.

Скомпілюйте пакет:

```
make
```

Цей пакет має набір тестів, але його можна запустити тільки після встановлення пакета. Тести знаходяться в каталозі `test/`. Детальнішу інформацію дивіться у файлі `README` в цьому каталозі.

Встановлення цього пакета замінить `libncursesw.so.6.5` на місці. Це може привести до збою процесу оболонки, який використовує код і дані з файлу бібліотеки. Встановіть пакет за допомогою `DESTDIR` і правильно замініть файл бібліотеки за допомогою команди `install` (заголовок `curses.h` також редагується, щоб забезпечити використання широкого символьного ABI, як ми це зробили в розділі 6.3, «Ncurses-6.5»):

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/libncursesw.so.6.5 /usr/lib
rm -v dest/usr/lib/libncursesw.so.6.5
sed -e 's/^#if.*XOPEN.*$/#if 1/' \
-i dest/usr/include/curses.h
cp -av dest/* /
```

Багато програм все ще очікують, що компілятор зможе знайти бібліотеки Ncurses, що не підтримують широкі символи. Змусьте такі програми з'єднуватися з бібліотеками, що підтримують широкі символи, за допомогою символьних посилань (зверніть увагу, що посилання .so є безпечними лише у випадку, якщо файл `curses.h` відредактовано так, щоб завжди використовувати ABI з широкими символами):

```
for lib in ncurses form panel menu ; do
    ln -sfv lib${lib}w.so /usr/lib/lib${lib}.so
    ln -sfv ${lib}w.pc      /usr/lib/pkgconfig/${lib}.pc
done
```

На звершення, переконайтесь, що старі програми, які шукають `-lcurses` під час компіляції, все ще можна компілювати:

```
ln -sfv libncursesw.so /usr/lib/libcurses.so
```

За бажанням, встановіть документацію Ncurses:

```
cp -v -R doc -T /usr/share/doc/ncurses-6.5
```



Note

Вищезазначені інструкції не створюють бібліотек Ncurses, що не підтримують широкі символи, оскільки жоден пакет, встановлений шляхом компіляції з вихідних кодів, не буде пов'язуватися з ними під час виконання. Однак єдині відомі бінарні програми, що пов'язуються з бібліотеками Ncurses, які не підтримують широкі символи, вимагають версії 5. Якщо вам необхідні такі бібліотеки через наявність бінарних програм або для відповідності LSB, перекомпілюйте пакет за допомогою наступних команд:

```
make distclean
./configure      --prefix=/usr          \
                  --with-shared        \
                  --without-normal     \
                  --without-debug       \
                  --without-cxx-binding \
                  --with-abi-version=5
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

8.30.2. Вміст Ncurses

Встановлені програми:	captoinfo (посилання на tic), clear, infocmp, infotocap (посилання на tic), ncursesw6-config, reset (посилання на tset), tabs, tic, toe, tput, and tset
Встановлені бібліотеки:	libcurses.so (символічне посилання), libform.so (символічне посилання), libformw.so, libmenu.so (символічне посилання), libmenuw.so,

libcurses.so (symbolічне посилання), libcursesw.so, libcurses++w.so, libpanel.so (symbolічне посилання), and libpanelw.so

Встановлені директорії:

/usr/share/tabset, /usr/share/terminfo та /usr/share/doc/ncurses-6.5

Короткі описи

captoinfo	Перетворює опис termcap в опис terminfo.
clear	Очищає екран, якщо це можливо.
infocmp	Порівнює або виводить описи terminfo.
infotocap	Перетворює опис terminfo в опис termcap.
ncursesw6-config	Надає інформацію про конфігурацію для ncurses.
reset	Перезавантажує термінал до його значень за замовчуванням.
tabs	Очищає і встановлює табуляцію на терміналі.
tic	Компілятор опису записів terminfo, який перетворює файл terminfo з вихідного формату у двійковий формат, необхідний для процедур бібліотеки ncurses [Файл terminfo містить інформацію про можливості певного терміналу].
toe	Перелічує всі доступні типи терміналів, надаючи основну назву та опис для кожного з них.
tput	Робить значення можливостей, що залежать від терміналу, доступними для оболонки; також може використовуватися для скидання або ініціалізації терміналу або повідомлення його повної назви.
tset	Може використовуватися для ініціалізації терміналів
libcursesw	Містить функції для відображення тексту на екрані терміналу різними складними способами; хорошим прикладом використання цих функцій є меню, що відображається під час виконання команди make menuconfig
libcurses++w	Містить зв'язки C++ для інших бібліотек у цьому пакеті
libformw	Містить функції для реалізації форм
libmenuw	Містить функції для реалізації меню
libpanelw	Містить функції для реалізації панелей

8.31. Sed-4.9

Пакет Sed містить редактор потоків.

Приблизний час побудови: 0.3 SBU

Необхідний простір на диску: 30 MB

8.31.1. Інсталяція Sed

Підготуйте Sed до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет і створіть HTML-документацію:

```
make
make html
```

Щоб перевірити результати, виконайте:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Встановіть пакет та документацію:

```
make install
install -d -m755      /usr/share/doc/sed-4.9
install -m644 doc/sed.html /usr/share/doc/sed-4.9
```

8.31.2. Вміст Sed

Встановлені програми: sed

Встановлені директорії: /usr/share/doc/sed-4.9

Короткі описи

sed	Фільтрує та перетворює текстові файли за один прохід
------------	--

8.32. Psmisc-23.7

Пакет Psmisc містить програми для відображення інформації про запущені процеси.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 6.7 MB

8.32.1. Інсталяція Psmisc

Підготуйте Psmisc до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.32.2. Вміст Psmisc

Встановлені програми: fuser, killall, peekfd, prtstat, pslog, pstree, and pstree.x11 (посилання на pstree)

Короткі описи

fuser	Повідомляє ідентифікатори процесів (PID) процесів, які використовують задані файли або файлові системи.
killall	Вбиває процеси за іменем; надсилає сигнал усім процесам, що виконують будь-яку із заданих команд.
peekfd	Переглядає дескриптори файлів запущеного процесу, заданого за його PID.
prtstat	Виводить інформацію про процес.
pslog	Повідомляє про поточний шлях журналів процесу
pstree	Відображає запущені процеси у вигляді дерева
pstree.x11	Те саме, що й pstree, за винятком того, що перед виходом чекає на підтвердження

8.33. Gettext-0.24

Пакет Gettext містить утиліти для інтернаціоналізації та локалізації. Вони дозволяють компілювати програми з підтримкою NLS (Native Language Support), що дає змогу виводити повідомлення рідною мовою користувача.

Приблизний час побудови: 1.7 SBU

Необхідний простір на диску: 290 MB

8.33.1. Інсталяція Gettext

Підготуйте Gettext до компіляції:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.24
```

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

8.33.2. Вміст Gettext

Встановлені програми:	autopoint, envsubst, gettext, gettext.sh, gettextize, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin, and xgettext
Встановлені бібліотеки:	libasprintf.so, libgettextlib.so, libgettextpo.so, libgettextsrc.so, libtextstyle.so, та preloadable_libintl.so
Встановлені директорії:	/usr/lib/gettext, /usr/share/doc/gettext-0.24, /usr/share/gettext, and /usr/share/gettext-0.24

Короткі описи

autopoint	Копіює стандартні файли інфраструктури Gettext у пакет джерел
envsubst	Замінює змінні середовища у рядках формату оболонки
gettext	Перекладає повідомлення природною мовою на мову користувача, шукаючи переклад у каталозі повідомлень
gettext.sh	В основному служить бібліотекою функцій оболонки для gettext
gettextize	Копіює всі стандартні файли Gettext у вказаний каталог верхнього рівня пакета, щоб розпочати його інтернаціоналізацію

msgattrib	Фільтрує повідомлення каталогу перекладів за їхніми атрибутами та маніпулює атрибутами
msgcat	Об'єднує та зливає вказані файли .po
msgcmp	Порівнює два файли .po, щоб перевірити, чи містять вони одинаковий набір рядків msgid
msgcomm	Знаходить повідомлення, які є спільними для заданих файлів .po.
msgconv	Перетворює каталог перекладів у іншу кодування символів.
msgen	Створює каталог перекладів англійською мовою.
msgexec	Застосовує команду до всіх перекладів каталогу перекладів.
msgfilter	Застосовує фільтр до всіх перекладів каталогу перекладів.
msgfmt	Генерує бінарний каталог повідомень з каталогу перекладів.
msggrep	Витягує всі повідомлення каталогу перекладів, які відповідають заданому шаблону або належать до певних вихідних файлів
msginit	Створює новий файл .po, ініціалізуючи метаінформацію значеннями з середовища користувача
msgmerge	Об'єднує два необроблені переклади в один файл
msgunfmt	Декомпілює бінарний каталог повідомень в необроблений текст перекладу
msguniq	Уніфікує дублікати перекладів в каталогі перекладів
ngettext	Відображає переклади текстового повідомлення рідною мовою, граматична форма якого залежить від числа
recode-sr-latin	Перекодує сербський текст з кирилиці в латиницю
xgettext	Витягує рядки повідомень, що підлягають перекладу, з заданих вихідних файлів, щоб створити перший шаблон перекладу
<i>libasprintf</i>	Визначає клас <i>autosprintf</i> , який робить процедури виводу у форматі C придатними для використання в програмах C++, для використання з рядками <string> та потоками <iostream>
<i>libgettextlib</i>	Містить загальні процедури, що використовуються різними програмами Gettext; вони не призначені для загального використання
<i>libgettextpo</i>	Використовується для написання спеціалізованих програм, що обробляють файли .po; ця бібліотека використовується, коли стандартних програм, що постачаються з Gettext (таких як msgcomm, msgcmp, msgattrib та msgen), недостатньо
<i>libgettextsrc</i>	Надає загальні процедури, що використовуються різними програмами Gettext; вони не призначені для загального використання
<i>libtextstyle</i>	Бібліотека стилів тексту

preloadable_libintl

Бібліотека, призначена для використання LD_PRELOAD, яка допомагає libintl реєструвати неперекладені повідомлення

8.34. Bison-3.8.2

Пакет Bison містить генератор синтаксичного аналізатора.

Приблизний час побудови: 2.1 SBU

Необхідний простір на диску: 62 MB

8.34.1. Інсталяція Bison

Підготуйте Bison до компіляції:

```
./configure --prefix=/usr --docdir=/usr/share/doc/bison-3.8.2
```

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.34.2. Вміст Bison

Встановлені програми: bison та yacc

Встановлені бібліотеки: liby.a

Встановлені директорії: /usr/share/bison

Короткі описи

bison Генерує, на основі низки правил, програму для аналізу структури текстових файлів; Bison є заміною для Yacc (Yet Another Compiler Compiler)

yacc Оболонка для **bison**, призначена для програм, які все ще викликають **yacc** замість **bison**; вона викликає **bison** з опцією **-y**

liby Бібліотека Yacc, що містить реалізації сумісних з Yacc функцій **yyerror** та **main**; ця бібліотека зазвичай не дуже корисна, але POSIX вимагає її наявності

8.35. Grep-3.11

Пакет Grep містить програми для пошуку по вмісту файлів.

Приблизний час побудови: 0.4 SBU

Необхідний простір на диску: 39 MB

8.35.1. Інсталяція Grep

Спочатку видаліть попередження про використання egrep та fgrep, яке призводить до невдачі тестів деяких пакетів:

```
sed -i "s/echo/#echo/" src/egrep.sh
```

Підготуйте Grep до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.35.2. Вміст Grep

Встановлені програми: egrep, fgrep та grep

Короткі описи

egrep Виводить рядки, що відповідають розширеному регулярному виразу. Ця команда застаріла, замість неї використовуйте **grep -E**.

fgrep Виводить рядки, що відповідають списку фікованих рядків. Ця команда застаріла, замість неї використовуйте **grep -F**

grep Виводить рядки, що відповідають базовому регулярному виразу.

8.36. Bash-5.2.37

Пакет Bash містить оболонку Bourne-Again Shell.

Приблизний час побудови: 1.4 SBU

Необхідний простір на диску: 53 MB

8.36.1. Інсталяція Bash

Підготуйте Bash до компіляції:

```
./configure --prefix=/usr \
            --without-bash-malloc \
            --with-installed-readline \
            --docdir=/usr/share/doc/bash-5.2.37
```

Значення нових конфігураційних параметрів:

--with-installed-readline

Ця опція вказує Bash використовувати бібліотеку readline, яка вже встановлена в системі, замість власної версії readline.

Скомпілюйте пакет:

```
make
```

Перейдіть до «Встановіть пакет», якщо не виконуєте набір тестів.

Щоб підготувати тести, переконайтесь, що користувач-тестувальник має права на запис у дерево sources:

```
chown -R tester .
```

Набір тестів цього пакету призначений для виконання користувачем, який не є root і володіє терміналом, підключеним до стандартного вводу. Щоб задовольнити цю вимогу, створіть новий псевдотермінал за допомогою Expect і запустіть тести як користувач tester:

```
su -s /usr/bin/expect tester << "EOF"
set timeout -1
spawn make tests
expect eof
lassign [wait] _ _ _ value
exit $value
EOF
```

Набір тестів використовує diff для виявлення різниці між результатами виконання тестового скрипта та очікуваними результатами. Будь-який результат від diff (з префіксом < and >) вказує на невдачу тесту, якщо тільки немає повідомлення про те, що різницю можна ігнорувати. Відомо, що один тест під назвою run-builtins не проходить на деяких дистрибутивах хостів із різницею в першому рядку результату.

Встановіть пакет:

```
make install
```

Запустіть новоскладену програму bash (замість тієї, яка виконується в даний момент):

```
exec /usr/bin/bash --login
```

8.36.2. Вміст Bash

Встановлені програми: bash, bashbug, and sh (link to bash)

Встановлені директорії: /usr/include/bash, /usr/lib/bash та /usr/share/doc/bash-5.2.37

Короткі описи

bash	Широко використовуваний інтерпретатор команд; він виконує багато типів розширень і підставлень у наданому командному рядку перед її запуском, що робить цей інтерпретатор потужним інструментом
bashbug	Скрипт оболонки, що допомагає користувачеві складати та надсилати електронною поштою стандартні форматовані звіти про помилки, що стосуються bash
sh	Символічне посилання на програму bash ; коли викликається як sh , bash намагається якомога точніше імітувати поведінку запуску історичних версій sh , одночасно відповідаючи стандарту POSIX

8.37. Libtool-2.5.4

Пакет Libtool містить скрипт підтримки загальної бібліотеки GNU. Він спрощує використання спільних бібліотек за допомогою послідовного, портативного інтерфейсу.

Приблизний час побудови: 0.6 SBU

Необхідний простір на диску: 44 MB

8.37.1. Інсталяція Libtool

Підготуйте Libtool до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

Видаліть непотрібну статичну бібліотеку:

```
rm -fv /usr/lib/libltdl.a
```

8.37.2. Вміст Libtool

Встановлені програми: libtool та libtoolize

Встановлені бібліотеки: libltdl.so

Встановлені директорії: /usr/include/libltdl та /usr/share/libtool

Короткі описи

libtool

Надає загальні послуги з підтримки створення бібліотек.

libtoolize

Надає стандартний спосіб додавання підтримки libtool до пакета.

libltdl

Приховує різні складнощі відкриття динамічно завантажених бібліотек

8.38. GDBM-1.24

Пакет GDBM містить GNU Database Manager. Це бібліотека функцій бази даних, яка використовує розширене хешування і працює як стандартний UNIX dbm. Бібліотека надає примітиви для зберігання пар ключ/дані, пошуку і вилучення даних за їх ключем, а також видалення ключа разом з даними.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 13 MB

8.38.1. Інсталяція GDBM

Підготуйте GDBM до компіляції:

```
./configure --prefix=/usr \
            --disable-static \
            --enable-libgdbm-compat
```

Значення конфігураційних параметрів:

--enable-libgdbm-compat

Цей перемикач увімкне компіляцію бібліотеки сумісності libgdbm. Деякі пакети поза LFS можуть потребувати старіших процедур DBM, які вона надає.

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.38.2. Вміст GDBM

Встановлені програми: gdbm_dump, gdbm_load та gdbmtool

Встановлені бібліотеки: libgdbm.so та libgdbm_compat.so

Короткі описи

gdbm_dump

Зберігає базу даних GDBM у файл.

gdbm_load

Відтворює базу даних GDBM із файлу дампа.

gdbmtool

Тестує та модифікує базу даних GDBM.

libgdbm

Містить функції для роботи з хешованою базою даних.

libgdbm_compat

Бібліотека сумісності, що містить стари функції DBM.

8.39. Gperf-3.1

Gperf генерує ідеальну хеш-функцію з набору ключів.

Приблизний час побудови: меншу ніж 0.1 SBU

Необхідний простір на диску: 6.1 MB

8.39.1. Інсталляція Gperf

Підготуйте Gperf до компіляції:

```
./configure --prefix=/usr --docdir=/usr/share/doc/gperf-3.1
```

Скомпілюйте пакет:

```
make
```

Відомо, що тести не проходять, якщо виконувати кілька тестів одночасно (опція `-j` більше 1). Щоб тестиувати результати, виконайте:

```
make -j1 check
```

Встановіть пакет:

```
make install
```

8.39.2. Вміст Gperf

Встановлені програми: gperf

Встановлені директорії: /usr/share/doc/gperf-3.1

Короткі описи

gperf

Генерує ідеальну хеш-функцію з набору ключів.

8.40. Expat-2.6.4

Пакет Expat містить потокоорієнтовану бібліотеку С для аналізу XML.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 14 MB

8.40.1. Інсталяція Expat

Підготуйте Expat до компіляції:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.6.4
```

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

За бажанням встановіть документацію:

```
install -v -m644 doc/*.{html,css} /usr/share/doc/expat-2.6.4
```

8.40.2. Вміст Expat

Встановлені програми: xmlwf

Встановлені бібліотеки: libexpat.so

Встановлені директорії: /usr/share/doc/expat-2.6.4

Короткі описи

xmlwf	Невалідаційна утиліта, для перевірки, чи є XML-документи правильно сформованими.
libexpat	Містить функції API для аналізу XML

8.41. Inetutils-2.6

Пакет Inetutils містить програми для базової роботи в мережі.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 35 MB

8.41.1. Інсталяція Inetutils

Спочатку створіть пакет за допомогою gcc-14.1 або пізнішої версії:

```
sed -i 's/def HAVE_TERMCAP_TGETENT/ 1/' telnet/telnet.c
```

Підготуйте Inetutils до компіляції:

```
./configure --prefix=/usr \
--bindir=/usr/bin \
--localstatedir=/var \
--disable-logger \
--disable-whois \
--disable-rcp \
--disable-rexec \
--disable-rlogin \
--disable-rsh \
--disable-servers
```

Значення конфігураційних параметрів:

--disable-logger

Ця опція запобігає встановленню Inetutils програми logger, яка використовується скриптами для передачі повідомлень до System Log Daemon. Не встановлюйте її, оскільки Util-linux встановлює більш нову версію.

--disable-whois

Ця опція вимикає збірку клієнта Inetutils whois, який є застарілим. Інструкції щодо кращого клієнта whois знаходяться в книзі BLFS.

--disable-r*

Ці параметри вимикають збірку застарілих програм, які не слід використовувати через проблеми з безпекою. Функції, що надаються цими програмами, можуть бути надані пакетом openssh в книзі BLFS.

--disable-servers

Це вимикає встановлення різних мережевих серверів, що входять до складу пакета Inetutils. Ці сервери вважаються невідповідними для базової системи LFS. Деякі з них є небезпечними за свою природою і вважаються безпечними лише в довірених мережах. Зверніть увагу, що для багатьох з цих серверів доступні кращі замінники.

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

Перемістіть програму у відповідне місце:

```
mv -v /usr/{,s}bin/ifconfig
```

8.41.2. Вміст Inetutils

Встановлені програми: dnsdomainname, ftp, ifconfig, hostname, ping, ping6, talk, telnet, tftp та traceroute

Короткі описи

dnsdomainname	Показати доменне ім'я DNS системи.
ftp	Програма протоколу передачі файлів.
hostname	Показує або встановлює ім'я хоста.
ifconfig	Керує мережевими інтерфейсами.
ping	Відправляє пакети запиту ехо і повідомляє, скільки часу займає відповідь.
ping6	Версія ping для мереж IPv6
talk	Використовується для спілкування з іншим користувачем
telnet	Інтерфейс до протоколу TELNET
tftp	Проста програма для передачі файлів
traceroute	Відстежує маршрут, яким проходять ваші пакети від хоста, на якому ви працюєте, до іншого хоста в мережі, показуючи всі проміжні переходи (шлюзи) на цьому шляху

8.42. Less-668

Пакет Less містить програму для перегляду текстових файлів.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 14 MB

8.42.1. Інсталяція Less

Підготуйте Less до компіляції:

```
./configure --prefix=/usr --sysconfdir=/etc
```

Значення конфігураційних параметрів:

--sysconfdir=/etc

Ця опція вказує програмам, створеним пакетом, шукати конфігураційні файли в /etc.

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.42.2. Вміст Less

Встановлені програми: less, lessecho та lesskey

Короткі описи

less

Програма для перегляду файлів або пейджер; вона відображає вміст заданого файлу, дозволяючи користувачеві прокручувати, знаходити рядки та переходити до позначок

lessecho

Необхідно для розширення метасимволів, таких як * та ?, в іменах файлів у системах Unix

lesskey

Використовується для визначення комбінацій клавіш для less

8.43. Perl-5.40.1

Пакет Perl містить мову Practical Extraction and Report Language.

Приблизний час побудови: 1.3 SBU

Необхідний простір на диску: 245 MB

8.43.1. Інсталяція Perl

Ця версія Perl створює модулі Compress::Raw::Zlib та Compress::Raw::BZip2. За замовчуванням Perl використовуватиме внутрішню копію джерел для створення. Виконайте наступну команду, щоб Perl використовував бібліотеки, встановлені в системі:

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

Щоб мати повний контроль над налаштуванням Perl, ви можете видалити опції «-des» з наступної команди та вручну вибрati способ побудови цього пакета. Або ж скористайтесь командою, як показано нижче, щоб використовувати стандартні налаштування, які Perl визначає автоматично:

```
sh Configure -des \
-D prefix=/usr \
-D vendorprefix=/usr \
-D privlib=/usr/lib/perl5/5.40/core_perl \
-D archlib=/usr/lib/perl5/5.40/core_perl \
-D sitelib=/usr/lib/perl5/5.40/site_perl \
-D sitearch=/usr/lib/perl5/5.40/site_perl \
-D vendorlib=/usr/lib/perl5/5.40/vendor_perl \
-D vendorarch=/usr/lib/perl5/5.40/vendor_perl \
-D man1dir=/usr/share/man/man1 \
-D man3dir=/usr/share/man/man3 \
-D pager="/usr/bin/less -isR" \
-D useshrplib \
-D usethreads
```

Значення конфігураційних параметрів:

-D pager="/usr/bin/less -isR"

Це гарантує, що замість **more** буде використовуватися **less**.

-D man1dir=/usr/share/man/man1 -D man3dir=/usr/share/man/man3

Оскільки Groff ще не встановлено, **Configure** не створюватиме сторінки man для Perl. Ці параметри замінюють цю поведінку.

-D usethreads

Скомпілюйте Perl з підтримкою потоків.

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
TEST_JOBS=$ (nproc) make test_harness
```

Встановіть пакет очітіті:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

8.43.2. Вміст Perl

Встановлені програми:	corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.40.1 (жорстке посилання на perl), perlbug, perldoc, perlivp, perlthunks (жорстке посилання на perlbug), piconv, pl2pm, pod2html, pod2man, pod2text, pod2usage, podchecker, podselect, prove, ptar, ptardiff, ptargrep, shasum, splain, xsubpp, and zipdetails
Встановлені бібліотеки:	Багато з них неможливо перелічити тут усі
Встановлені директорії:	/usr/lib/perl5

Короткі описи

corelist	Командний рядок для Module::CoreList
cpan	Взаємодія з Comprehensive Perl Archive Network (CPAN) з командного рядка
enc2xs	Створює розширення Perl для модуля Encode з Unicode Character Mappings або Tcl Encoding Files
encguess	Вгадує тип кодування одного або декількох файлів
h2ph	Конвертує файли заголовків .h C у файли заголовків .ph Perl
h2xs	Конвертує файли заголовків .h C у розширення Perl
instmodsh	Скрипт оболонки для перевірки встановлених модулів Perl; може створювати архів tarball із встановленого модуля
json_pp	Конвертує дані між певними форматами вводу та виводу
libnetcfg	Може використовуватися для налаштування модуля libnet Perl
perl	Поєднує деякі з найкращих функцій C, sed , awk та sh в єдину мову Swiss Army
perl5.40.1	Жорстке посилання на perl
perlbug	Використовується для створення звітів про помилки в Perl або модулях, що входять до його складу, та надсилання їх поштою
perldoc	Відображає частину документації у форматі pod, вбудовану в дерево інсталяції Perl або в скрипт Perl
perlivp	Процедура перевірки інсталяції Perl; може використовуватися для перевірки правильності інсталяції Perl та його бібліотек
perlthunks	Використовується для створення повідомлень з подякою, які надсилаються розробникам Perl

piconv	Версія Perl конвертера кодування символів iconv
pl2pm	Простий інструмент для конвертації файлів Perl4 .pl у модулі Perl5 .pm
pod2html	Конвертує файли з формату pod у формат HTML
pod2man	Конвертує дані pod у форматований вхід *roff
pod2text	Конвертує дані pod у форматований текст ASCII
pod2usage	Виводить повідомлення про використання з вбудованих документів pod у файлах
podchecker	Перевіряє синтаксис файлів документації у форматі pod
podselect	Відображає вибрані розділи документації pod
prove	Командний рядок для виконання тестів модуля Test::Harness
ptar	Програма, схожа на tar , написана на Perl
ptardiff	Програма Perl, яка порівнює розпакований архів з нерозпакованим
ptargrep	Програма Perl, яка застосовує зіставлення шаблонів до вмісту файлів в архіві tar.
shasum	Виводить або перевіряє контрольні суми SHA.
splain	Використовується для примусового виведення докладних діагностичних попереджень в Perl.
xsubpp	Перетворює код Perl XS в код C.
zipdetails	Відображає детальну інформацію про внутрішню структуру файлу Zip.

8.44. XML::Parser-2.47

Модуль XML::Parser є інтерфейсом Perl до XML-парсера Джеймса Кларка, Expat.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 2.4 MB

8.44.1. Інсталяція XML::Parser

Підготуйте XML::Parser до компіляції:

```
perl Makefile.PL
```

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.44.2. Вміст XML::Parser

Встановлені модулі: Expat.so

Короткі описи

Expat

надає інтерфейс Perl Expat

8.45. Intltool-0.51.0

Модуль XML::Parser є інтерфейсом Perl до XML-парсера Джеймса Кларка, Expat.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 1.5 MB

8.45.1. Інсталяція Intltool

Спочатку виправте попередження, яке викликається perl-5.22 та пізнішими версіями:

```
sed -i 's:\\\\${:\\\\$\\\\{:' intltool-update.in
```



Note

Вищезазначений регулярний вираз виглядає незвично через велику кількість символів зворотного слеша. Він додає зворотний слеш перед правим фігурним дужкою в послідовності “\${”, в результаті чого отримуємо “\$\{”.

Підготуйте Intltool до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
install -v -Dm644 doc/I18N-HOWTO /usr/share/doc/intltool-0.51.0/I18N-HOWTO
```

8.45.2. Вміст Intltool

Встановлені програми: intltool-extract, intltool-merge, intltool-prepare, intltool-update та intltoolize

Встановлені директорії: /usr/share/doc/intltool-0.51.0 та /usr/share/intltool

Короткі описи

intltoolize	Готує пакет для використання intltool.
intltool-extract	Генерує файли заголовків, які можуть бути прочитані gettext .
intltool-merge	Об'єднує перекладені рядки в різni типи файлів.
intltool-prepare	Оновлює файли pot і об'єднує їх з файлами перекладу.
intltool-update	Оновлює файли шаблонів po і об'єднує їх з перекладами.

8.46. Autoconf-2.72

Пакет Autoconf містить програми для створення скриптів оболонки, які можуть автоматично налаштовувати вихідний код.

Приблизний час побудови: менше ніж 0.1 SBU (блізько 0.4 SBU разом з тестами)

Необхідний простір на диску: 25 MB

8.46.1. Інсталяція Autoconf

Підготуйте Autoconf до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.46.2. Вміст Autoconf

Встановлені програми: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, ifnames

Встановлені директорії: /usr/share/autoconf

Короткі описи

autoconf

Створює скрипти оболонки, які автоматично налаштовують пакети вихідного коду програмного забезпечення для адаптації до багатьох типів Unix-подібних систем; скрипти налаштування, які він створює, є незалежними — для їх виконання не потрібна програма **autoconf**

autoheader

Інструмент для створення файлів-шаблонів операторів C `#define` для використання в `configure`

autom4te

Оболонка для макропроцесора M4

autoreconf

Автоматично запускає **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize** та **libtoolize** у правильному порядку, щоб заощадити час при внесенні змін до файлів шаблонів **autoconf** та **automake**

autoscan

Допомагає створити файл `configure.in` для програмного пакета; він перевіряє вихідні файли в дереві каталогів, шукаючи в них типові проблеми з переносимістю, і створює файл `configure.scan`, який служить попереднім файлом `configure.in` для пакета

autoupdate

Змінює файл `configure.in`, який все ще викликає макроси **autoconf** за їх старими іменами, щоб використовувати поточні імена макросів

ifnames

Допомагає при написанні файлів `configure.in` для програмного пакету; виводить ідентифікатори, які пакет використовує в умовних операторах препроцесора С [Якщо пакет вже налаштований для певної переносимості, ця програма може допомогти визначити, що потрібно перевірити **configure**. Вона також може заповнити прогалини у файлі `configure.in`, згенерованому **autoscan**.]

8.47. Automake-1.17

Модуль XML::Parser є інтерфейсом Perl до XML-парсера Джеймса Кларка, Expat.

Приблизний час побудови: менше ніж 0.1 SBU (блізько 1.1 SBU разом з тестами)

Необхідний простір на диску: 121 MB

8.47.1. Інсталяція Automake

Підготуйте Automake до компіляції:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.17
```

Скомпілюйте пакет:

```
make
```

Використання чотирьох паралельних завдань прискорює тестування навіть на системах з меншою кількістю логічних ядер через внутрішні затримки в окремих тестах. Щоб перевірити результати, виконайте:

```
make -j$(($nproc>4?$nproc:4)) check
```

Замініть \$(...) на кількість логічних ядер, які ви хочете використовувати, якщо не хочете використовувати всі.

Встановіть пакет:

```
make install
```

8.47.2. Вміст Automake

Встановлені програми: aclocal, aclocal-1.17 (жорстке посилання з aclocal), automake, automake-1.17 (жорстке посилання з automake)

Встановлені директорії: /usr/share/aclocal-1.17, /usr/share/automake-1.17, /usr/share/doc/automake-1.17

Короткі описи

aclocal	Генерує файли aclocal.m4 на основі вмісту файлів configure.in
aclocal-1.17	Жорстке посилання на aclocal
automake	Інструмент для автоматичного генерування файлів Makefile.in з файлів Makefile.am [Щоб створити всі файли Makefile.in для пакета, запустіть цю програму в каталозі верхнього рівня. Скануючи файл configure.in, вона автоматично знаходить кожен відповідний файл Makefile.am і генерує відповідний файл Makefile.in.]
automake-1.17	Жорстке посилання на automake

8.48. OpenSSL-3.4.1

Пакет OpenSSL містить інструменти управління та бібліотеки, пов'язані з криптографією. Вони корисні для надання криптографічних функцій іншим пакетам, таким як OpenSSH, програмам електронної пошти та веб-браузери (для доступу до сайтів HTTPS).

Приблизний час побудови: 1.8 SBU

Необхідний простір на диску: 920 MB

8.48.1. Інсталяція OpenSSL

Підготуйте OpenSSL до компіляції:

```
./config --prefix=/usr \
          --openssldir=/etc/ssl \
          --libdir=lib \
          shared \
          zlib-dynamic
```

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
HARNESS_JOBS=$ (nproc) make test
```

Відомо, що один тест, 30-test_afalg.t, завершується з помилкою, якщо в ядрі хоста не ввімкнено CONFIG_CRYPTO_USER_API_SKCIPHER або немає опцій, що забезпечують реалізацію AES з CBC (наприклад, комбінація CONFIG_CRYPTO_AES і CONFIG_CRYPTO_CBC, або CONFIG_CRYPTO_AES_NI_INTEL, якщо процесор підтримує AES-NI). Якщо він завершується з помилкою, його можна безпечно ігнорувати.

Встановіть пакет:

```
sed -i '/INSTALL_LIBS/s/libcrypto.a libssl.a/' Makefile
make MANSUFFIX=ssl install
```

Додайте версію до імені каталогу документації, щоб забезпечити узгодженість з іншими пакетами:

```
mv -v /usr/share/doc/openssl /usr/share/doc/openssl-3.4.1
```

За бажанням, встановіть додаткову документацію:

```
cp -vfr doc/* /usr/share/doc/openssl-3.4.1
```

**Note**

Ви повинні оновити OpenSSL, коли буде оголошено про випуск нової версії, яка виправляє вразливості. Починаючи з OpenSSL 3.0.0, схема версій OpenSSL відповідає формату MAJOR.MINOR.PATCH. Сумісність API/ABI гарантується для одного і того ж номера версії MAJOR. Оскільки LFS встановлює тільки спільні бібліотеки, немає необхідності перекомпілювати пакети, які пов'язані з `libcrypto.so` або `libssl.so`, *при оновленні до версії з тим самим номером MAJOR.*

Однак, будь-які запущені програми, пов'язані з цими бібліотеками, потрібно зупинити і перезапустити. Детальніше читайте у відповідних розділах у розділі 8.2.1, «Проблеми оновлення».

8.48.2. Вміст OpenSSL

Встановлені програми:	<code>c_rehash</code> та <code>openssl</code>
Встановлені бібліотеки:	<code>libcrypto.so</code> та <code>libssl.so</code>
Встановлені директорії:	<code>/etc/ssl</code> , <code>/usr/include/openssl</code> , <code>/usr/lib/engines</code> та <code>/usr/share/doc/openssl-3.4.1</code>

Короткі описи

<code>c_rehash</code>	є Perl-скриптом, який сканує всі файли в каталозі та додає символічні посилання на їх хеш-значення. Використання <code>c_rehash</code> вважається застарілим і має бути замінено командою <code>openssl rehash</code>
<code>openssl</code>	є командним рядком для використання різних криптографічних функцій криптобібліотеки OpenSSL з оболонки. Він може використовуватися для різних функцій, які описані в <code>openssl(1)</code>
<code>libcrypto.so</code>	реалізує широкий спектр криптографічних алгоритмів, що використовуються в різних інтернет-стандартах. Послуги, що надаються цією бібліотекою, використовуються в реалізаціях OpenSSL для SSL, TLS і S/ MIME, а також використовуються для реалізації OpenSSH, OpenPGP та інших криптографічних стандартів
<code>libssl.so</code>	реалізує протокол Transport Layer Security (TLS v1). Він надає багатий API, документація про який знаходиться в <code>ssl(7)</code>

8.49. Libelf із Elfutils-0.192

Libelf — це бібліотека для обробки файлів ELF (Executable and Linkable Format).

Приблизний час побудови: 0.3 SBU

Необхідний простір на диску: 135 MB

8.49.1. Інсталяція Libelf

Libelf є частиною пакета elfutils-0.192. Використовуйте файл `elfutils-0.192.tar.bz2` як вихідний архів.

Підготуйте Libelf до компіляції:

```
./configure --prefix=/usr \
            --disable-debuginfod \
            --enable-libdebuginfod=dummy
```

Скомпілюйте пакет:

make

Щоб запустити набір тестів, виконайте:

make check

Встановіть тільки Libelf:

```
make -C libelf install  
install -vm644 config/libelf.pc /usr/lib/pkgconfig  
rm /usr/lib/libelf.a
```

8.49.2. Вміст Libelf

Встановлена бібліотека: libelf.so

Встановлена директорія: /usr/include/elfutils

Короткі описи

libelf.so

Містить функції API для обробки об'єктів файлів ELF.

8.50. Libffi-3.4.7

Бібліотека Libffi надає портативний інтерфейс програмування високого рівня для різних конвенцій виклику. Це дозволяє програмісту викликати будь-яку функцію, визначену описом інтерфейсу виклику, під час виконання.

FFI означає Foreign Function Interface (інтерфейс зовнішніх функцій). FFI дозволяє програмі, написаній на одній мові, викликати програму, написану на іншій мові. Зокрема, Libffi може забезпечити міст між інтерпретатором, таким як Perl або Python, і підпрограмами спільної бібліотеки, написаними на С або С++.

Приблизний час побудови: 1.7 SBU

Необхідний простір на диску: 11 MB

8.50.1. Інсталяція Libffi



Note

Як і GMP, Libffi компілюється з оптимізаціями, специфічними для використовуваного процесора. Якщо ви компілюєте для іншої системи, змініть значення параметра `--with-gcc-arch`= у наступній команді на назву архітектури, повністю реалізовану процесором цієї системи. Якщо цього не зробити, всі програми, що посилаються на libffi, будуть викликати помилки незаконної операції.

Підготуйте Libffi до компіляції:

```
./configure --prefix=/usr \
            --disable-static \
            --with-gcc-arch=native
```

Значення конфігураційних параметрів:

`--with-gcc-arch=native`

Забезпечує оптимізацію GCC для поточної системи. Якщо це не вказано, система визначається автоматично, і згенерований код може бути некоректним. Якщо згенерований код буде скопійовано з нативної системи на систему з меншими можливостями, використовуйте систему з меншими можливостями як параметр. Детальніше про альтернативні типи систем див. опції x86 у посібнику GCC.

Скомпілюйте пакет:

```
make
```

Щоб запустити набір тестів, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.50.2. Вміст Libffi

Встановлена бібліотека: libffi.so

Короткі описи

libffi

Містить функції API інтерфейсу зовнішніх функцій

8.51. Python-3.13.2

Пакет Python 3 містить середовище розробки Python. Він корисний для об'єктно-орієнтованого програмування, написання скриптів, створення прототипів великих програм і розробки цілих додатків. Python — це інтерпретована комп'ютерна мова.

Приблизний час побудови: 2.1 SBU

Необхідний простір на диску: 501 MB

8.51.1. Інсталяція Python

Підготуйте Python до компіляції:

```
./configure --prefix=/usr \
            --enable-shared \
            --with-system-expat \
            --enable-optimizations
```

Значення конфігураційних параметрів:

`--with-system-expat`

Цей параметр увімкне зв'язування з системною версією Expat.

`--enable-optimizations`

Цей параметр увімкне розширені, але трудомісткі кроки оптимізації. Інтерпретатор буде побудовано двічі; тести, виконані на першій збірці, будуть використані для поліпшення оптимізованої кінцевої версії.

Скомпілюйте пакет:

```
make
```

Відомо, що деякі тести іноді зависають на невизначений час. Тому, щоб перевірити результати, запустіть набір тестів, але встановіть 2-хвилинний ліміт часу для кожного тестового випадку:

```
make test TESTOPTS="--timeout 120"
```

Для відносно повільної системи може знадобитися збільшити обмеження за часом, і 1 SBU (виміряно під час побудови Binutils pass 1 з одним ядром процесора) має бути достатньо. Деякі тести є нестабільними, тому набір тестів автоматично повторно запускає тести, які не пройшли. Якщо тест не пройшов, але потім пройшов при повторному запуску, його слід вважати пройденим. Відомо, що один тест, `test_ssl`, не проходить у середовищі chroot.

Встановіть пакет:

```
make install
```

У цій книзі ми використовуємо команду `pip3` для встановлення програм і модулів Python 3 для всіх користувачів як `root` у декількох місцях. Це суперечить рекомендаціям розробників Python: встановлювати пакети у віртуальне середовище або у домашній каталог звичайного користувача (запустивши `pip3` як цей користувач). Кожного разу, коли `pip3` запускається користувачем `root`, з'являється багаторядкове попередження.

Головною причиною такої рекомендації є уникнення конфліктів із системним менеджером пакетів (наприклад, `dpkg`). LFS не має системного менеджера пакетів, тому це не є проблемою. Крім того, `pip3` перевірятиме наявність нової версії себе кожного разу, коли запускається. Оскільки розпізнавання доменних

імен ще не налаштоване в середовищі LFS chroot, **pip3** не може перевірити наявність нової версії себе і видасть попередження.

Після завантаження системи LFS і налаштування мережевого з'єднання буде видано інше попередження, яке повідомить користувача про необхідність оновлення **pip3** з попередньо скомпільованого колеса на PyPI (коли буде доступна нова версія). Але LFS вважає **pip3** частиною Python 3, тому його не слід оновлювати окремо. Крім того, оновлення з попередньо скомпільованого колеса буде відхилятися від нашої мети: створити систему Linux з вихідного коду. Тому попередження про нову версію **pip3** також слід ігнорувати. Якщо бажаєте, ви можете придушити всі ці попередження, виконавши наступну команду, яка створює файл конфігурації:

```
cat > /etc/pip.conf << EOF
[global]
root-user-action = ignore
disable-pip-version-check = true
EOF
```



Important

У LFS та BLFS ми зазвичай створюємо та встановлюємо модулі Python за допомогою команди **pip3**. Переконайтесь, що команди **pip3 install** в обох книгах виконуються як користувач `root` (якщо це не стосується віртуального середовища Python). Виконання **pip3 install** як користувач, що не є `root`, може здаватися ефективним, але це призведе до того, що встановлений модуль буде недоступним для інших користувачів.

pip3 install не перевстановлює вже встановлений модуль автоматично. При використанні команди **pip3 install** для оновлення модуля (наприклад, з `meson-0.61.3` до `meson-0.62.0`), вставте опцію `--upgrade` у командний рядок. Якщо дійсно необхідно понизити версію модуля або перевстановити ту саму версію з якихось причин, вставте `--force-reinstall --no-deps` у командний рядок.

За бажанням, встановіть попередньо відформатовану документацію:

```
install -v -dm755 /usr/share/doc/python-3.13.2/html
tar --strip-components=1 \
    --no-same-owner \
    --no-same-permissions \
    -C /usr/share/doc/python-3.13.2/html \
    -xvf ../python-3.13.2-docs-html.tar.bz2
```

Значення команд інсталяції документації:

`--no-same-owner` та `--no-same-permissions`

Переконайтесь, що встановлені файли мають правильні права власності та доступи. Без цих опцій tar встановить файли пакета із значеннями, заданими творцем.

8.51.2. Вміст Python

Встановлені програми: 2to3, idle3, pip3, pydoc3, python3, python3-config

Встановлені бібліотеки: libpython3.13.so та libpython3.so

Встановлені директорії: /usr/include/python3.13, /usr/lib/python3, /usr/share/doc/python-3.13.2

Короткі описи

2to3

це програма на Python, яка читує вихідний код Python 2.x і застосовує низку виправлень, щоб перетворити його на дійсний код Python 3.x

idle3

це скрипт-обгортка, який відкриває графічний редактор, що підтримує Python. Щоб цей скрипт працював, ви повинні встановити Tk перед Python, щоб було створено модуль Tkinter Python.

pip3

Інсталятор пакетів для Python. Ви можете використовувати pip для інсталяції пакетів з Python Package Index та інших індексів.

pydoc3

це інструмент документації Python

python3

є інтерпретатором для Python, інтерпретованої, інтерактивної, об'єктно-орієнтованої мови програмування.

8.52. Flit-Core-3.11.0

Flit-core — це компоненти Flit (інструменту для пакування простих модулів Python), що відповідають за побудову дистрибутивів.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 1.0 MB

8.52.1. Інсталяція Flit-Core

Зберірте пакунок:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Встановіть пакет:

```
pip3 install --no-index --find-links dist flit_core
```

Значення рір3 конфігураційних параметрів та команд:

wheel

Ця команда створює архів wheel для цього пакета.

-w dist

Вказує рір помістити створений архів wheel у каталог dist.

--no-cache-dir

Забороняє рір копіювати створений архів wheel у каталог /root/.cache/pip.

install

Ця команда встановлює пакет.

--no-build-isolation, --no-deps та --no-index

Ці опції запобігають завантаженню файлів з онлайн-репозиторію пакетів (PyPI). Якщо пакети встановлені в правильному порядку, рір не потрібно завантажувати жодних файлів; ці опції додають певну безпеку на випадок помилки користувача.

--find-links dist

Вказує рір шукати архіви wheel у каталозі dist.

8.52.2. Вміст Flit-Core

Встановлені директорії: /usr/lib/python3.13/site-packages/flit_core та
/usr/lib/python3.13/site-packages/flit_core-3.11.0.dist-info

8.53. Wheel-0.45.1

Wheel — це бібліотека Python, яка є еталонною реалізацією стандарту пакування Python wheel.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 1.6 MB

8.53.1. Інсталяція Wheel

Скомпілюйте Wheel за допомогою наступної команди:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Встановіть Wheel за допомогою наступної команди:

```
pip3 install --no-index --find-links dist wheel
```

8.53.2. Вміст Wheel

Встановлена програма: wheel

Встановлені директорії: /usr/lib/python3.13/site-packages/wheel та
/usr/lib/python3.13/site-packages/wheel-0.45.1.dist-info

Короткі описи

wheel є утилітою для розпакування, пакування або конвертації архівів wheel.

8.54. Setuptools-75.8.1

Setuptools — це інструмент, який використовується для завантаження, створення, встановлення, оновлення та видалення пакетів Python.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 26 MB

8.54.1. Інсталяція Setuptools

Зберіть пакет:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Встановіть пакет:

```
pip3 install --no-index --find-links dist setuptools
```

8.54.2. Вміст Setuptools

Встановлені директорії: /usr/lib/python3.13/site-packages/_distutils_hack, /usr/lib/python3.13/site-packages/pkg_resources, /usr/lib/python3.13/site-packages/setuptools, /usr/lib/python3.13/site-packages/setuptools-75.8.1.dist-info

8.55. Ninja-1.12.1

Ninja — це невелика система збірки, орієнтована на швидкість.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 37 MB

8.55.1. Інсталяція Ninja

Під час виконання **ninja** зазвичай використовує максимально можливу кількість паралельних процесів. За замовчуванням це кількість ядер у системі плюс два. Це може привести до перегріву процесора або вичерпання пам'яті системи. Коли **ninja** викликається з командного рядка, передача параметра **-jN** обмежить кількість паралельних процесів. Деякі пакети вбудовують виконання **ninja** і не передають йому параметр **-j**.

Використання наведеної нижче **опціональної** процедури дозволяє користувачеві обмежити кількість паралельних процесів за допомогою змінної середовища **NINJAJOBS**. **Наприклад**, налаштування:

```
export NINJAJOBS=4
```

обмежить **ninja** чотирма паралельними процесами.

За бажанням, змусьте **ninja** розпізнавати змінну середовища **NINJAJOBS**, запустивши редактор потоку:

```
sed -i '/int Guess/a \
    int j = 0; \
    char* jobs = getenv( "NINJAJOBS" ); \
    if ( jobs != NULL ) j = atoi( jobs ); \
    if ( j > 0 ) return j; \
' src/ninja.cc
```

Зберіть Ninja за допомогою:

```
python3 configure.py --bootstrap --verbose
```

Значення параметрів збірки:

--bootstrap

Цей параметр змушує Ninja перекомпілювати себе для поточної системи.

--verbose

Цей параметр змушує **configure.py** показувати хід компіляції Ninja.

Тести пакета не можуть виконуватися в середовищі chroot. Для цього потрібна програма **stake**. Але основна функція цього пакета вже перевірена шляхом його перекомпіляції (з опцією **--bootstrap**).

Встановіть пакет:

```
install -vM755 ninja /usr/bin/
install -vDm644 misc/bash-completion /usr/share/bash-completion/completions/ninja
install -vDm644 misc/zsh-completion /usr/share/zsh/site-functions/_ninja
```

8.55.2. Вміст Ninja

Встановлені програми: **ninja**

Короткі описи

ninja

Система збірки Ninja

8.56. Meson-1.7.0

Meson — це система компіляції з відкритим кодом, розроблена для забезпечення максимальної швидкості та зручності використання.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 44 MB

8.56.1. Інсталяція Meson

Скомпілюйте Meson за допомогою наступної команди:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Набір тестів вимагає деяких пакетів, що виходять за межі LFS.

Встановіть пакет:

```
pip3 install --no-index --find-links dist meson
install -vDm644 data/shell-completions/bash/meson/usr/share/bash-completion/completions/meson
install -vDm644 data/shell-completions/zsh/_meson /usr/share/zsh/site-functions/_meson
```

Значення параметрів інсталяції:

-w dist

Поміщає створені wheels в каталог dist.

--find-links dist

Встановлює wheels з каталогу dist.

8.56.2. Вміст Meson

Встановлена програма: meson

Встановлені директорії: /usr/lib/python3.13/site-packages/meson-1.7.0.dist-info та /usr/lib/python3.13/site-packages/mesonbuild

Короткі описи

meson	Високопродуктивна система збірки
--------------	----------------------------------

8.57. Kmod-34

Пакет Kmod містить бібліотеки та утиліти для завантаження модулів ядра.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 11 MB

8.57.1. Інсталяція Kmod

Підготуйте Kmod до компіляції:

```
mkdir -p build
cd      build

meson setup --prefix=/usr .. \
             --sbindir=/usr/sbin \
             --buildtype=release \
             -D manpages=false
```

Значення параметрів конфігурації:

-D manpages=false

Ця опція вимикає генерацію сторінок man, для яких потрібна зовнішня програма.

Скомпілюйте пакет:

```
ninja
```

Набір тестів цього пакету вимагає raw заголовків ядра (а не «sanitized» заголовків ядра, встановлених раніше), які виходять за межі LFS.

Тепер встановіть пакет:

```
ninja install
```

8.57.2. Вміст Kmod

Встановлена програма: depmod (посилання на kmod), insmod (посилання на kmod), kmod, lsmod (посилання на kmod), modinfo (посилання на kmod), modprobe (посилання на kmod), and rmmod (посилання на kmod)

Встановлені бібліотеки: libkmod.so

Короткі описи

depmod	Створює файл залежностей на основі символів, знайдених в існуючому наборі модулів; цей файл залежностей використовується modprobe для автоматичного завантаження необхідних модулів.
insmod	Встановлює завантажуваний модуль в працююче ядро.
kmod	Завантажує та вивантажує модулі ядра.
lsmod	Виводить список завантажених модулів.
modinfo	Перевіряє об'єктний файл, пов'язаний з модулем ядра, і відображає всю інформацію, яку може зібрати
modprobe	Використовує файл залежностей, створений depmod , для автоматичного

завантаження відповідних модулів

rmmod

Вивантажує модулі з працючого ядра

libkmod

Ця бібліотека використовується іншими програмами для завантаження та вивантаження модулів ядра

8.58. Coreutils-9.6

Пакет Coreutils містить основні утиліти, необхідні для кожної операційної системи.

Приблизний час побудови: 1.2 SBU

Необхідний простір на диску: 182 MB

8.58.1. Інсталляція Coreutils

POSIX вимагає, щоб програми з Coreutils правильно розпізнавали межі символів навіть у багатобайтових локалях. Наступний патч виправляє цю невідповідність та інші помилки, пов'язані з інтернаціоналізацією.

```
patch -Np1 -i ../../coreutils-9.6-i18n-1.patch
```



Note

У цьому патчі було виявлено багато помилок. Перед тим, як повідомляти про нові помилки розробникам Coreutils, будь ласка, перевірте спочатку, чи можна відтворити ці помилки без цього патча.

Тепер підготуйте Coreutils до компіляції:

```
autoreconf -fv
automake -af
FORCE_UNSAFE_CONFIGURE=1 ./configure \
    --prefix=/usr \
    --enable-no-install-program=kill,uptime
```

Значення команд та конфігураційних параметрів:

autoreconf -fv

Патч для інтернаціоналізації змінив систему збірки, тому конфігураційні файли необхідно перегенерувати. Зазвичай ми використовуємо опцію `-i` для оновлення стандартних допоміжних файлів, але для цього пакета вона не працює, оскільки `configure.ac` вказав стару версію `gettext`.

automake -af

Допоміжні файли `automake` не були оновлені `autoreconf` через відсутність опції `-i`. Ця команда оновлює їх, щоб запобігти збою збірки.

`FORCE_UNSAFE_CONFIGURE=1`

Ця змінна середовища дозволяє збирати пакет користувачем `root`.

`--enable-no-install-program=kill,uptime`

Мета цього параметра — запобігти встановленню Coreutils програм, які будуть встановлені іншими пакетами.

Скомпілюйте пакет:

```
make
```

Перейдіть до розділу «Встановити пакет», якщо не виконуєте набір тестів.

Тепер набір тестів готовий до виконання. Спочатку виконайте тести, які призначені для виконання як користувач `root`:

```
make NON_ROOT_USERNAME=tester check-root
```

Ми будемо виконувати решту тестів як користувач `tester`. Деякі тести вимагають, щоб користувач був членом більш ніж однієї групи. Щоб ці тести не були пропущені, додайте тимчасову групу і додайте до неї користувача `tester`:

```
groupadd -g 102 dummy -U tester
```

Виправте деякі дозволи, щоб користувач, який не є суперкористувачем, міг компілювати та запускати тести:

```
chown -R tester .
```

Тепер запустіть тести (використовуючи `/dev/null` для стандартного вводу, інакше два тести можуть бути порушенні, якщо LFS будеться в графічному терміналі або сесії в SSH або GNU Screen, оскільки стандартний ввід підключений до PTY з дистрибутива хоста, а до вузла пристрою для такого PTY не можна отримати доступ з середовища chroot LFS):

```
su tester -c "PATH=$PATH make -k RUN_EXPENSIVE_TESTS=yes check" \
< /dev/null
```

Видаліть тимчасову групу:

```
groupdel dummy
```

Встановіть пакет:

```
make install
```

Перемістити програми в місця, визначені FHS:

```
mv -v /usr/bin/chroot /usr/sbin
mv -v /usr/share/man/man1/chroot.1 /usr/share/man/man8/chroot.8
sed -i 's/"1"/"8"/' /usr/share/man/man8/chroot.8
```

8.58.2. Вміст Coreutils

Встановлені програми: [
], b2sum, base32, base64, basename, basenc, cat, chcon, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, numfmt, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami, yes

Встановлені бібліотеки: libstdbuf.so (in /usr/libexec/coreutils)

Встановлені директорії: /usr/libexec/coreutils

Короткі описи

[

Це фактична команда, /usr/bin/[; вона є синонімом команди **test**

base32

Кодує та декодує дані відповідно до специфікації base32 (RFC 4648)

base64

Кодує та декодує дані відповідно до специфікації base64 (RFC 4648)

b2sum

Виводить або перевіряє контрольні суми BLAKE2 (512-бітні)

basename	Видаляє будь-який шлях та заданий суфікс з імені файлу
basenc	Кодує або декодує дані за допомогою різних алгоритмів
cat	Об'єднує файли у стандартний вивід
chcon	Змінює контекст безпеки для файлів і каталогів
chgrp	Змінює групову власність файлів і каталогів
chmod	Змінює права доступу до кожного файла на заданий режим; режим може бути або символічним представленням змін, що мають бути зроблені, або вісімковим числом, що представляє нові права доступу
chown	Змінює власника файлів і каталогів (користувача та/або групу)
chroot	Виконує команду з вказаним каталогом як каталогом /
cksum	Виводить контрольну суму циклічного надлишкового кодування (CRC) та кількість байтів кожного вказаного файла
comm	Порівнює два відсортовані файли, виводячи в трьох стовпцях рядки, які є унікальними, та рядки, які є спільними
cp	Копіює файли.
csplit	Розділяє заданий файл на кілька нових файлів, розділяючи їх за заданими шаблонами або номерами рядків, і виводить кількість байтів кожного нового файла.
cut	Виводить частини рядків, вибираючи частини за заданими полями або позиціями.
date	Відображає поточну дату та час у заданому форматі або встановлює системну дату та час
dd	Копіює файл, використовуючи заданий розмір блоку та кількість, при цьому за бажанням виконуючи перетворення
df	Повідомляє про кількість вільного (та використаного) дискового простору на всіх змонтованих файлових системах або тільки на файлових системах, що містять вибрані файли
dir	Перелічує вміст кожного заданого каталогу (так само, як команда ls)
dircolors	Виводить команди для встановлення змінної середовища LS_COLORS , щоб змінити колірну схему, яка використовується командою ls
dirname	Витягує частину (частини) каталогу із заданого імені (імен)
du	Повідомляє про обсяг дискового простору, який використовується поточним каталогом, кожним із заданих каталогів (включаючи всі підкаталоги) або кожним із заданих файлів
echo	Відображає задані рядки.
env	Виконує команду в модифікованому середовищі.

expand	Перетворює табуляції в пробіли.
expr	Оцінює вирази.
factor	Виводить прості множники заданих цілих чисел.
false	Не виконує ніяких дій, завершується з кодом статусу, що вказує на невдачу.
fmt	Переформатує абзаци в заданих файлах.
fold	Переносить рядки в заданих файлах.
groups	Повідомляє про членство користувача в групах.
head	Виводить перші десять рядків (або задану кількість рядків) кожного заданого файла.
hostid	Повідомляє про числовий ідентифікатор (у шістнадцятковій системі числення) хоста.
id	Повідомляє про ефективний ідентифікатор користувача, ідентифікатор групи та членство в групах поточного користувача або заданого користувача.
install	Копіює файли, встановлюючи їхні режими дозволів і, якщо можливо, їхнього власника та групу.
join	Об'єднує рядки, що мають однакові поля з'єднання, з двох окремих файлів.
link	Створює жорстке посилання (з вказаним іменем) на файл
In	Створює жорсткі або м'які (symbolічні) посилання між файлами
logname	Повідомляє ім'я користувача, що ввійшов у систему
ls	Виводить вміст кожного вказаного каталогу
md5sum	Повідомляє або перевіряє контрольні суми Message Digest 5 (MD5)
mkdir	Створює каталоги з заданими іменами.
mkfifo	Створює FIFO (First-In, First-Out), «іменовані канали» в термінології UNIX, з заданими іменами.
mknod	Створює вузли пристріїв з заданими іменами; вузол пристрію — це спеціальний символічний файл, спеціальний блоковий файл або FIFO.
mktemp	Створює тимчасові файли безпечним способом; використовується в скриптах.
mv	Переміщує або перейменовує файли або каталоги.
nice	Запускає програму зі зміненим пріоритетом планування
nl	Пронумерує рядки з заданих файлів
nohup	Запускає команду, стійку до зависань, з перенаправленням її виводу в файл журналу

nproc	Виводить кількість одиниць обробки, доступних для процесу
numfmt	Перетворює числа в рядки, зрозумілі для людини, або навпаки
od	Виводить файли в вісімковому та інших форматах
paste	Об'єднує задані файли, послідовно з'єднуючи відповідні рядки поруч, розділені символами табуляції
pathchk	Перевіряє, чи є імена файлів дійсними або портативними
pinky	Є легким клієнтом finger; він повідомляє деяку інформацію про заданих користувачів
pr	Розбиває файли на сторінки та стовпці для друку
printenv	Виводить середовище
printf	Виводить задані аргументи відповідно до заданого формату, подібно до функції printf мови С.
ptx	Створює переставлений індекс із вмісту заданих файлів, з кожним ключовим словом у його контексті.
pwd	Повідомляє ім'я поточного робочого каталогу.
readlink	Повідомляє значення заданого символічного посилання.
realpath	Виводить вирішений шлях.
rm	Видаляє файли або каталоги.
rmdir	Видаляє каталоги, якщо вони порожні.
runcon	Виконує команду із заданим контекстом безпеки.
seq	Виводить послідовність чисел у заданому діапазоні та із заданим кроком.
sha1sum	Виводить або перевіряє 160-бітні контрольні суми алгоритму Secure Hash Algorithm 1 (SHA1).
sha224sum	Виводить або перевіряє 224-бітні контрольні суми алгоритму Secure Hash Algorithm.
sha256sum	Виводить або перевіряє 256-бітні контрольні суми алгоритму Secure Hash Algorithm.
sha384sum	Виводить або перевіряє 384-бітні контрольні суми алгоритму Secure Hash Algorithm.
sha512sum	Виводить або перевіряє 512-бітні контрольні суми алгоритму Secure Hash Algorithm
shred	Перезаписує задані файли повторно складними візерунками, ускладнюючи відновлення даних
shuf	Перемішує рядки тексту
sleep	Зупиняється на заданий проміжок часу

sort	Сортує рядки із заданих файлів
split	Розділяє заданий файл на частини за розміром або за кількістю рядків
stat	Відображає стан файлу або файлової системи.
stdbuf	Виконує команди зі зміненими операціями буферизації для своїх стандартних потоків.
stty	Встановлює або повідомляє налаштування термінальної лінії.
sum	Виводить контрольну суму та кількість блоків для кожного заданого файлу.
sync	Очищає буфери файлової системи; примусово записує змінені блоки на диск та оновлює суперблок.
tac	Об'єднує задані файли у зворотному порядку.
tail	Виводить останні десять рядків (або задану кількість рядків) кожного заданого файлу.
tee	Зчитує зі стандартного вводу, записуючи одночасно у стандартний вивід та у задані файли.
test	Порівнює значення та перевіряє типи файлів.
timeout	Виконує команду з обмеженням за часом.
touch	Змінює часові мітки файлів, встановлюючи час доступу та модифікації заданих файлів на поточний час; файли, які не існують, створюються з нульовою довжиною
tr	Перекладає, стискає та видаляє задані символи зі стандартного вводу
true	Нічого не робить, успішно; завжди завершується з кодом статусу, що вказує на успіх
truncate	Стискає або розширює файл до вказаного розміру
tsort	Виконує топологічне сортування; записує повністю впорядкований список відповідно до часткового впорядкування у вказаному файлі
tty	Повідомляє ім'я файлу терміналу, підключенного до стандартного вводу
uname	Повідомляє системну інформацію
unexpand	Перетворює пробіли на табуляції
uniq	Відкидає всі, крім одного, послідовні однакові рядки
unlink	Видаляє вказаний файл
users	Повідомляє імена користувачів, які зараз увійшли в систему
vdir	Те саме, що ls -l
wc	Повідомляє кількість рядків, слів і байтів для кожного заданого файлу, а

також загальні суми, якщо задано більше ніж один файл

who
Повідомляє, хто ввійшов у систему

whoami
Повідомляє ім'я користувача, пов'язане з поточним ефективним ідентифікатором користувача

yes
Повторно виводить у або заданий рядок, доки не буде зупинено

libstdbuf
Бібліотека, що використовується stdbuf

8.59. Check-0.15.2

Check — це фреймворк для модульного тестування на мові C.

Приблизний час побудови: 0.1 SBU (блізько 2.1 SBU із тестами)

Необхідний простір на диску: 11 MB

8.59.1. Інсталяція Check

Підготуйте Check до компіляції:

```
./configure --prefix=/usr --disable-static
```

Скомпілюйте пакет:

```
make
```

Компіляція зараз завершена. Щоб запустити набір тестів Check, виконайте наступну команду:

```
make check
```

Встановіть пакет:

```
make docdir=/usr/share/doc/check-0.15.2 install
```

8.59.2. Вміст Check

Встановлені програми: checkmk

Встановлені бібліотеки: libcheck.so

Короткі описи

checkmk

Скрипт Awk для генерації модульних тестів С для використання з фреймворком модульного тестування Check

libcheck.so

Містить функції, що дозволяють викликати Check з тестової програми

8.60. Diffutils-3.11

Пакет Diffutils містить програми, які показують відмінності між файлами або каталогами.

Приблизний час побудови: 0.4 SBU

Необхідний простір на диску: 50 MB

8.60.1. Інсталяція Diffutils

Підготуйте Diffutils до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.60.2. Вміст Diffutils

Встановлені програми: cmp, diff, diff3, sdiff

Короткі описи

cmp

Порівнює два файли і повідомляє про будь-які відмінності байт за байтом

diff

Порівнює два файли або каталоги і повідомляє, які рядки в файлах відрізняються

diff3

Порівнює три файли рядок за рядком

sdiff

Об'єднує два файли і інтерактивно виводить результати

8.61. Gawk-5.3.1

Пакет Gawk містить програми для роботи з текстовими файлами.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 43 MB

8.61.1. Інсталяція Gawk

Спочатку переконайтесь, що деякі непотрібні файли не встановлені:

```
sed -i 's/extras//' Makefile.in
```

Підготуйте Gawk до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Встановіть пакет:

```
rm -f /usr/bin/gawk-5.3.1
make install
```

Значення команд:

rm -f /usr/bin/gawk-5.3.1

Система збірки не буде відтворювати жорстке посилання `gawk-5.3.1`, якщо воно вже існує. Видаліть його, щоб переконатися, що попереднє жорстке посилання, встановлене в розділі 6.9, «Gawk-5.3.1», оновлено тут.

Процес інсталяції вже створив `awk` як символічне посилання на `gawk`, створіть також його сторінку та як символічне посилання:

```
ln -sv gawk.1 /usr/share/man/man1/awk.1
```

За бажанням встановіть документацію:

```
install -vDm644 doc/{awkforai.txt,*.{eps,pdf,jpg}} -t /usr/share/doc/gawk-5.3.1
```

8.61.2. Вміст Gawk

Встановлені програми: awk (посилання на gawk), gawk, and gawk-5.3.1

Встановлені бібліотеки: filefuncs.so, fnmatch.so, fork.so, inplace.so, intdiv.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoWay.so, rwaray.so, time.so (все в /usr/lib/gawk)

Встановлені директорії: /usr/lib/gawk, /usr/libexec/awk, /usr/share/awk, /usr/share/doc/gawk-5.3.1

Короткі описи

awk

Посилання на **gawk**

gawk

Програма для обробки текстових файлів; це реалізація **awk** для GNU.

gawk-5.3.1

Жорстке посилання на **gawk**

8.62. Findutils-4.10.0

Пакет Findutils містить програми для пошуку файлів. Програми призначені для пошуку серед усіх файлів у дереві каталогів та для створення, обслуговування і пошуку в базі даних (часто швидше, ніж рекурсивний пошук, але ненадійно, якщо база даних не оновлювалася нещодавно). Findutils також надає програму **xargs**, яку можна використовувати для виконання вказаної команди над кожним файлом, вибраним під час пошуку.

Приблизний час побудови: 0.7 SBU

Необхідний простір на диску: 63 MB

8.62.1. Інсталляція Findutils

Підготуйте Findutils до компіляції:

```
./configure --prefix=/usr --localstatedir=/var/lib/locate
```

Значення конфігураційних параметрів:

--localstatedir

Ця опція переміщує базу даних **locate** до /var/lib/locate, що є місцем розташування, яке відповідає FHS.

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Встановіть пакет:

```
make install
```

8.62.2. Вміст Findutils

Встановлені програми: find, locate, updatedb, xargs

Встановлені директорії: /var/lib/locate

Короткі описи

find Шукає у вказаних деревах каталогів файли, що відповідають заданим критеріям

locate Шукає у базі даних імен файлів і повідомляє імена, що містять заданий рядок або відповідають заданому шаблону

updatedb Оновлює базу даних locate; сканує всю файлову систему (включаючи інші файлові системи, що наразі підключенні, якщо не вказано інше) і вносить кожне знайдене ім'я файлу до бази даних

xargs Може використовуватися для застосування заданої команди до списку файлів

8.63. Groff-1.23.0

Пакет Groff містить програми для обробки та форматування тексту та зображень.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 108 MB

8.63.1. Інсталяція Groff

Groff очікує, що змінна середовища PAGE міститиме розмір паперу за замовчуванням. Для користувачів у Сполучених Штатах підходить PAGE=letter. В інших країнах може бути більш підходящим PAGE=A4. Хоча розмір паперу за замовчуванням налаштовується під час компіляції, його можна змінити пізніше, ввівши «A4» або «letter» у файл /etc/papersize.

Підготуйте Groff до компіляції:

```
PAGE=<paper_size> ./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Перевірте результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.63.2. Вміст Groff

Встановлені програми: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, glilypond, gperl, gpinyin, grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, gropdf, grops, grotty, hptftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfmom, pdfroff, pfbtops, pic, pic2graph, post-grohtml, preconv, pre-grohtml, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit, troff

Встановлені директорії: /usr/lib/groff, /usr/share/doc/groff-1.23.0, /usr/share/groff

Короткі описи

addftinfo

Читає файл шрифту troff і додає додаткову інформацію про метрику шрифту, яка використовується системою **groff**.

afmtodit

Створює файл шрифту для використання з **groff** і **grops**.

chem

Препроцесор Groff для створення діаграм хімічної структури.

eqn

Компілює описи рівнянь, вбудованих у вхідні файли troff, у команди, які розуміє **troff**.

eqn2graph

Перетворює troff EQN (рівняння) у обрізане зображення.

gdiffmk

Позначає відмінності між файлами groff/nroff/troff.

glilypond

Перетворює ноти, написані мовою lilypond, у мову groff.

gperl	Препроцесор для groff , що дозволяє вставляти код perl у файли groff
gpinyin	Препроцесор для groff , що дозволяє вставляти Pinyin (мандаринська китайська мова, написана латиницею) у файли groff .
grap2graph	Перетворює файл програми grap у обрізане растркове зображення (grap — це стара мова програмування Unix для створення діаграм)
grn	Препроцесор groff для файлів gremlin
grodvi	Драйвер для groff , який створює вихідні файли у форматі TeX dvi
groff	Інтерфейс для системи форматування документів groff ; зазвичай він запускає програму troff та постпроцесор, відповідний для вибраного пристрою
groffer	Відображає файли groff та сторінки man на терміналах X та tty
grog	Читає файли та визначає, які з опцій groff -e , -man , -me , -mm , -ms , -p , -s та -t необхідні для друку файлів, та повідомляє команду groff , включаючи ці опції
grolbp	Є драйвером groff для принтерів Canon CAPSL (лазерні принтери серій LBP-4 та LBP-8)
grolj4	Є драйвером для groff , який створює вихідні дані у форматі PCL5, придатному для принтера HP LaserJet 4
gropdf	Перетворює вихідні дані GNU troff у PDF
grops	Перетворює вихідні дані GNU troff у PostScript
grotty	Перетворює вихідні дані GNU troff у форму, придатну для пристрійв, подібних до друкарських машинок
hpftodit	Створює файл шрифту для використання з groff -Tlj4 з файлу метрики шрифту з тегами HP
indxbib	Створює інвертований індекс для бібліографічних баз даних із зазначенним файлом для використання з refer , lookbib та lkbib
lkbib	Шукає в бібліографічних базах даних посилання, що містять вказані ключі, і повідомляє про всі знайдені посилання
lookbib	Виводить підказку на стандартний потік помилок (якщо стандартний вхід не є терміналом), читає рядок, що містить набір ключових слів зі стандартного входу, шукає в бібліографічних базах даних у вказаному файлі посилання, що містять ці ключові слова, виводить всі знайдені посилання на стандартний вихід і повторює цей процес до кінця вводу
mmroff	Простий препроцесор для groff
neqn	Форматує рівняння для виводу в форматі American Standard Code for Information Interchange (ASCII)
nroff	Скрипт, що емулює команду nroff за допомогою groff

pdfmom	Є оболонкою навколо groff, що полегшує створення PDF-документів з файлів, відформатованих за допомогою макросів том.
pdfroff	Створює документи pdf за допомогою groff
pfbtops	Перетворює шрифт PostScript у форматі .pfb в ASCII
pic	Компілює описи зображень, вбудованих у вхідні файли troff або TeX, в команди, які розуміють TeX або troff
pic2graph	Перетворює діаграму PIC в обрізане зображення
post-grohtml	Перетворює вихідні дані GNU troff в HTML
preconv	Перетворює кодування вхідних файлів в щось, що розуміє GNU troff
pre-grohtml	Перетворює вихідні дані GNU troff в HTML
refer	Копіює вміст файлу до стандартного виводу, за винятком того, що рядки між .[і .] інтерпретуються як цитати, а рядки між .R1 і .R2 інтерпретуються як команди щодо того, як мають оброблятися цитати
roff2dvi	Перетворює файли roff у формат DVI.
roff2html	Перетворює файли roff у формат HTML.
roff2pdf	Перетворює файли roff у PDF.
roff2ps	Перетворює файли roff у файли ps.
roff2text	Перетворює файли roff у текстові файли.
roff2x	Перетворює файли roff в інші формати.
soelim	Читає файли і замінює рядки у форматі .so файлом, змістом згаданого файлу.
tbl	Компілює описи таблиць, вбудованих у вхідні файли troff, у команди, які розуміє troff
tfmtodit	Створює файл шрифту для використання з groff -Tdvi
troff	Має високу сумісність з Unix troff ; зазвичай його слід викликати за допомогою команди groff , яка також запускає препроцесори та постпроцесори у відповідному порядку та з відповідними опціями

8.64. GRUB-2.12

Пакет GRUB містить GRand Unified Bootloader (великий уніфікований завантажувач).

Приблизний час побудови: 0.3 SBU

Необхідний простір на диску: 166 MB

8.64.1. Інсталяція GRUB



Note

Якщо ваша система підтримує UEFI і ви бажаєте завантажувати LFS з UEFI, вам потрібно встановити GRUB з підтримкою UEFI (та його залежності), дотримуючись інструкцій на сторінці BLFS. Ви можете пропустити цей пакет або встановити цей пакет і пакет BLFS GRUB для UEFI без конфлікту (на сторінці BLFS наведено інструкції для обох випадків).



Warning

Скиньте всі змінні середовища, які можуть вплинути на збірку:

```
unset {C,CPP,CXX,LD}FLAGS
```

Не намагайтесь «налаштовувати» цей пакет за допомогою спеціальних пропорців компіляції. Цей пакет є завантажувачем. Низькорівневі операції в вихідному коді можуть бути порушені агресивною оптимізацією.

Додати файл, якого бракує у архіві релізу:

```
echo depends bli part_gpt > grub-core/extr_deps.1st
```

Підготуйте GRUB до компіляції:

```
./configure --prefix=/usr \
            --sysconfdir=/etc \
            --disable-efiemu \
            --disable-werror
```

Значення нових конфігураційних параметрів:

--disable-werror

Це дозволяє завершити збірку з попередженнями, введеними в більш нових версіях Flex.

--disable-efiemu

Ця опція мінімізує обсяг компіляції, вимикаючи деякі функції та виключаючи тестові програми, які не потрібні для LFS.

Скомпілюйте пакет:

```
make
```

Набір тестів для цього пакета не рекомендується. Більшість тестів залежать від пакетів, які недоступні в обмеженому середовищі LFS. Щоб все ж виконати тести, запустіть **make check**.

Встановіть пакет і перемістіть файл підтримки автодоповнення Bash у місце, рекомендоване розробниками автодоповнення Bash:

```
make install
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

Про те, як зробити вашу систему LFS завантажуваною за допомогою GRUB, йдеться в розділі 10.4, «Використання GRUB для налаштування процесу завантаження».

8.64.2. Вміст GRUB

Встановлені програми:	grub-bios-setup, grub-editenv, grub-file, grub-fstest, grub-glue-efi, grub-install, grub-kbdcomp, grub-macbless, grub-menu.lst2cfg, grub-mkconfig, grub-mkimage, grub-mklayout, grub-mknetdir, grub-mkpasswd-pbkdf2, grub-mkrelpath, grub-mkrescue, grub-mkstandalone, grub-ofpathname, grub-probe, grub-reboot, grub-render-label, grub-script-check, grub-set-default, grub-sparc64-setup, grub-syslinux2cfg
Встановлені директорії:	/usr/lib/grub, /etc/grub.d, /usr/share/grub, /boot/grub (коли grub-install запускається вперше)

Короткі описи

grub-bios-setup	Допоміжна програма для grub-install .
grub-editenv	Інструмент для редагування блоку середовища.
grub-file	Перевіряє, чи має вказаний файл заданий тип.
grub-fstest	Інструмент для налагодження драйвера файлової системи.
grub-glue-efi	Об'єднує 32-бітні та 64-бітні бінарні файли в один файл (для комп'ютерів Apple).
grub-install	Встановлює GRUB на ваш диск.
grub-kbdcomp	Скрипт, який перетворює розкладку xkb у розкладку, що розпізнається GRUB.
grub-macbless	Є благословенням у стилі Mac для файлових систем HFS або HFS+ (bless є особливістю комп'ютерів Apple; воно робить пристрій завантажувальним)
grub-menu.lst2cfg	Перетворює GRUB Legacy menu.lst у grub.cfg для використання з GRUB 2
grub-mkconfig	Генерує файл grub.cfg
grub-mkimage	Створює завантажувальний образ GRUB
grub-mklayout	Генерує файл розкладки клавіатури GRUB
grub-mknetdir	Готує каталог GRUB netboot
grub-mkpasswd-pbkdf2	Генерує зашифрований пароль PBKDF2 для використання в меню завантаження
grub-mkrelpath	Створює системний шлях відносно його кореня
grub-mkrescue	Створює завантажувальний образ GRUB, придатний для флоппі-диска,

	CDROM/DVD або USB-накопичувача
grub-mkstandalone	Генерує автономний образ
grub-ofpathname	Є допоміжною програмою, яка виводить шлях до пристрою GRUB
grub-probe	Перевіряє інформацію про пристрій для заданого шляху або пристрою
grub-reboot	Встановлює запис завантаження за замовчуванням для GRUB тільки для наступного завантаження
grub-render-label	Відображає Apple .disk_label для Apple Mac
grub-script-check	Перевіряє скрипт конфігурації GRUB на наявність синтаксичних помилок
grub-set-default	Встановлює запис завантаження за замовчуванням для GRUB
grub-sparc64-setup	Є допоміжною програмою для grub-setup
grub-syslinux2cfg	Перетворює файл конфігурації syslinux у формат grub.cfg

8.65. Gzip-1.13

Пакет Gzip містить програми для стиснення та розпакування файлів.

Приблизний час побудови: 0.3 SBU

Необхідний простір на диску: 21 MB

8.65.1. Інсталяція Gzip

Підготуйте Gzip до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.65.2. Вміст Gzip

Встановлені програми: gunzip, gzexe, gzip, uncompress (жорстке посилання з gunzip), zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore, and znew

Короткі описи

gunzip

Розпаковує файли у форматі gzip.

gzexe

Створює саморозпаковані виконувані файли.

gzip

Стискає вказані файли за допомогою кодування Lempel-Ziv (LZ77).

uncompress

Розпаковує стиснуті файли.

zcat

Розпаковує вказані файли у форматі gzip у стандартний вивід.

zcmp

Запускає cmp на файлах у форматі gzip.

zdiff

Запускає diff на gzip-файлах

zegrep

Запускає egrep на gzip-файлах

zfgrep

Запускає fgrep на gzip-файлах

zforce

Примусово додає розширення .gz до всіх заданих gzip-файлів, щоб gzip не стискає їх знову; це може бути корисно, коли імена файлів були обрізані під час передачі файлів

zgrep

Запускає grep на gzip-файлах

zless

Запускає less на gzip-файлах

zmore

Запускає more на gzip-файлах

znew

Перекомпресує файли з формату compress у формат gzip — з .z у .gz

8.66. IPRoute2-6.13.0

Пакет IPRoute2 містить програми для базової та розширеної роботи в мережах на основі IPV4.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 17 MB

8.66.1. Інсталяція IPRoute2

Програма **arpd**, що входить до цього пакету, не буде скомпільована, оскільки вона залежить від Berkeley DB, яка не встановлена в LFS. Однак каталог і сторінка довідки для **arpd** все одно будуть встановлені. Щоб цього уникнути, виконайте команди, наведені нижче.

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

Скомпілюйте пакет:

```
make NETNS_RUN_DIR=/run/netns
```

Цей пакет не має робочого набору тестів.

Встановіть пакет:

```
make SBINDIR=/usr/sbin install
```

За бажанням, встановіть документацію:

```
install -vDm644 COPYING README* -t /usr/share/doc/iproute2-6.13.0
```

8.66.2. Вміст IPRoute2

Встановлені програми: bridge, ctstat (посилання на Instat), genl, ifstat, ip, Instat, nstat, routel, rtacct, rtmon, rtpr, rtstat (посилання на Instat), ss, and tc

Короткі описи

bridge	Налаштовує мережеві мости
ctstat	Утиліта для перевірки стану з'єднання
genl	Універсальний інтерфейс утиліти netlink
ifstat	Відображає статистику інтерфейсу, включаючи кількість переданих і отриманих пакетів, за інтерфейсом
ip	Основний виконуваний файл. Він має кілька різних функцій, включаючи такі: ip link <device> дозволяє користувачам переглядати стан пристрій та вносити зміни ip addr дозволяє користувачам переглядати адреси та їх властивості, додавати нові адреси та видаляти старі ip neighbor дозволяє користувачам переглядати сусідські зв'язки та їх властивості, додавати нові сусідські записи та видаляти старі ip rule дозволяє користувачам переглядати політики маршрутизації та змінювати їх

ip route дозволяє користувачам переглядати таблицю маршрутизації та змінювати правила таблиці маршрутизації
ip tunnel дозволяє користувачам переглядати IP-тунелі та їх властивості, а також змінювати їх
ip maddr дозволяє користувачам переглядати мультикаст-адреси та їх властивості, а також змінювати їх
ip mroute дозволяє користувачам встановлювати, змінювати або видаляти мультикаст-маршрутизацію
ip monitor дозволяє користувачам постійно контролювати стан пристройів, адрес і маршрутів.

instat	Надає статистику мережі Linux; це узагальнена і більш функціональна заміна старої програми rtstat .
nstat	Відображає статистику мережі.
routel	Компонент ip route для переліку таблиць маршрутизації
rtacct	Відображає вміст /proc/net/rt_acct
rtmon	Утиліта моніторингу маршрутів
rtpr	Перетворює вихідні дані ip -o у читабельну форму
rtstat	Утиліта статусу маршруту
ss	Схожа на команду netstat ; показує активні з'єднання
tc	Контроль трафіку для реалізації Quality of Service (QoS) та Class of Service (CoS) tc qdisc дозволяє користувачам налаштовувати дисципліну черги tc class дозволяє користувачам налаштовувати класи на основі планування дисципліни черги tc filter дозволяє користувачам налаштовувати фільтрацію пакетів QoS/CoS tc monitor може використовуватися для перегляду змін, внесених до Traffic Control в ядрі.

8.67. Kbd-2.7.1

Пакет Kbd містить файли таблиць клавіш, шрифти консолі та утиліти клавіатури.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 34 MB

8.67.1. Інсталяція Kbd

Поведінка клавіш Backspace та Delete не є однаковою у всіх клавіатурних розкладках пакету Kbd. Наступний патч виправляє цю проблему для клавіатурних розкладок i386:

```
patch -Np1 -i ../../kbd-2.7.1-backspace-1.patch
```

Після виправлення клавіша Backspace генерує символ з кодом 127, а клавіша Delete генерує добре відому послідовність escape.

Видаліть зайву програму **resizecons** (вона вимагає неіснуючої бібліотеки **svgalib** для надання файлів відеорежimu — для нормального використання **setfont** розмірів шрифтів консолі) разом з її сторінкою довідки.

```
sed -i '/RESIZECONS_PROGS=/s/yes/no/' configure
sed -i 's/resizecons.8 //' docs/man/man8/Makefile.in
```

Підготуйте пакет Kbd до компіляції:

```
./configure --prefix=/usr --disable-vlock
```

Значення конфігураційних параметрів:

--disable-vlock

Ця опція запобігає компіляції утиліти vlock, оскільки вона вимагає бібліотеки PAM, яка недоступна в середовищі chroot.

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```



Note

Для деяких мов (наприклад, білоруської) пакет Kbd не надає корисної розкладки клавіатури, де стандартна розкладка «by» передбачає кодування ISO-8859-5, а зазвичай використовується розкладка CP1251. Користувачі таких мов повинні завантажувати робочі розкладки клавіатури окремо.

За бажанням, встановіть документацію:

```
cp -R -v docs/doc -T /usr/share/doc/kbd-2.7.1
```

8.67.2. Вміст Kbd

Встановлені програми:

chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (посилання на psfxtable), psfgettable (посилання на psfxtable), psfstriptable (посилання на psfxtable), psfxtable, setfont, setkeycodes, setleds, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start, unicode_stop

Встановлені директорії:

/usr/share/consolefonts, /usr/share/consoletrans, /usr/share/keymaps, /usr/share/doc/kbd-2.7.1, /usr/share/unimaps

Короткі описи

chvt	Змінює віртуальний термінал переднього плану.
dallocvt	Звільняє невикористані віртуальні термінали.
dumpkeys	Завантажує таблиці перекладу клавіатури.
fgconsole	Виводить номер активного віртуального терміналу.
getkeycodes	Виводить таблицю відповідності кодів сканування ядра та кодів клавіш.
kbdinfo	Отримує інформацію про стан консолі.
kbd_mode	Повідомляє або встановлює режим клавіатури.
kbdrate	Встановлює швидкість повторення та затримки клавіатури.
loadkeys	Завантажує таблиці перекладу клавіатури.
loadunimap	Завантажує таблицю відповідності Unicode та шрифтів ядра.
mapscrn	Застаріла програма, яка раніше завантажувала визначену користувачем таблицю відповідності символів виводу в драйвер консолі; зараз це робить setfont
openvt	Запускає програму на новому віртуальному терміналі (VT)
psfaddtable	Додає таблицю символів Unicode до шрифту консолі
psfgettable	Витягує вбудовану таблицю символів Unicode з шрифту консолі
psfstriptable	Видаляє вбудовану таблицю символів Unicode з шрифту консолі
psfxtable	Обробляє таблиці символів Unicode для шрифтів консолі
setfont	Змінює шрифти Enhanced Graphic Adapter (EGA) та Video Graphics Array (VGA) на консолі
setkeycodes	Завантажує записи таблиці відповідності сканованих кодів клавішам ядра; це корисно, якщо на клавіатурі є незвичайні клавіші
setleds	Встановлює пропорці клавіатури та світлодіоди (LED)
setmetamode	Визначає обробку метаклавіш клавіатури
setvtrgb	Встановлює карту кольорів консолі у всіх віртуальних терміналах

showconsolefont	Показує поточний шрифт екрану консолі EGA/VGA
showkey	Повідомляє скановані коди, коди клавіш та ASCII-коди клавіш, натиснутих на клавіатурі
unicode_start	Переводить клавіатуру та консоль у режим UNICODE [Не використовуйте цю програму, якщо ваш файл розкладки клавіатури не має кодування ISO-8859-1. Для інших кодувань ця утиліта видає неправильні результати.]
unicode_stop	Повертає клавіатуру та консоль із режиму UNICODE

8.68. Libpipeline-1.5.8

Пакет Libpipeline містить бібліотеку для гнучкого та зручного управління потоками підпроцесів.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 11 MB

8.68.1. Інсталяція Libpipeline

Підготуйте Libpipeline до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.68.2. Вміст Libpipeline

Встановлені бібліотеки: libpipeline.so

Короткі описи

`libpipeline`

Ця бібліотека використовується для безпечної побудови конвеєрів між підпроцесами.

8.69. Make-4.4.1

Пакет Make містить програму для управління створенням виконуваних файлів та інших файлів, що не є вихідними, пакета з вихідних файлів.

Приблизний час побудови: 0.7 SBU

Необхідний простір на диску: 13 MB

8.69.1. Інсталяція Make

Підготуйте Make до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Встановіть пакет:

```
make install
```

8.69.2. Вміст Make

Встановлені програми: make

Короткі описи

make	Автоматично визначає, які частини пакета потрібно (пере)компілювати, а потім видає відповідні команди
-------------	---

8.70. Patch-2.7.6

Пакет Patch містить програму для модифікації або створення файлів шляхом застосування файлу «патчу», який зазвичай створюється програмою **diff**.

Приблизний час побудови: 0.2 SBU

Необхідний простір на диску: 12 MB

8.70.1. Інсталяція Patch

Підготуйте Patch до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.70.2. Вміст Patch

Встановлені програми: patch

Короткі описи

patch	Змінює файли відповідно до файлу patch (Файл patch зазвичай є списком відмінностей, створеним за допомогою програми diff . Застосовуючи ці відмінності до оригінальних файлів, програма patch створює виправлені версії.)
--------------	---

8.71. Tar-1.35

Пакет Tar надає можливість створювати архіви tar, а також виконувати різні інші операції з архівами. Tar можна використовувати для раніше створених архівів, щоб витягувати файли, зберігати додаткові файли або оновлювати чи перелічувати файли, які вже були збережені.

Приблизний час побудови: 0.6 SBU

Необхідний простір на диску: 43 MB

8.71.1. Інсталяція Tar

Підготуйте Tar до компіляції:

```
FORCE_UNSAFE_CONFIGURE=1 \
./configure --prefix=/usr
```

Значення конфігураційних параметрів:

FORCE_UNSAFE_CONFIGURE=1

Це змушує тест для mknod виконуватися як root. Зазвичай вважається небезпечним виконувати цей тест як користувач root, але оскільки він виконується на системі, яка побудована лише частково, його можна замінити.

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Один тест, можливості: бінарне зберігання/відновлення, як відомо, не проходить, якщо його запустити, оскільки LFS не має selinux, але буде пропущений, якщо ядро хоста не підтримує розширені атрибути або мітки безпеки у файловій системі, що використовується для побудови LFS.

Встановіть пакет:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.35
```

8.71.2. Вміст Tar

Встановлені програми: tar

Встановлені директорії: /usr/share/doc/tar-1.35

Короткі описи

tar Створює, витягує файли та переглядає вміст архівів, також відомих як tar-архіви

8.72. Texinfo-7.2

Пакет Texinfo містить програми для читання, запису та перетворення інформаційних сторінок.

Приблизний час побудови: 0.3 SBU

Необхідний простір на диску: 160 MB

8.72.1. Інсталяція Texinfo

Підготуйте Texinfo до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

За бажанням, встановіть компоненти, що належать до інсталяції TeX:

```
make TEXMF=/usr/share/texmf install-tex
```

Значення конфігураційних параметрів:

TEXMF=/usr/share/texmf

Змінна makefile *TEXMF* містить розташування кореневого каталогу дерева TeX, якщо, наприклад, пакет TeX буде встановлено пізніше.

Система документації Info використовує звичайний текстовий файл для зберігання списку пунктів меню. Файл знаходиться в */usr/share/info/dir*. На жаль, через періодичні проблеми в Makefiles різних пакетів, іноді він може виходити з синхронізації з інформаційними сторінками, встановленими в системі. Якщо файл */usr/share/info/dir* потрібно перестворити, наступні опціональні команди виконують це завдання:

```
pushd /usr/share/info
rm -v dir
for f in *
do install-info $f dir 2>/dev/null
done
popd
```

8.72.2. Вміст Texinfo

Встановлені програми: info, install-info, makeinfo (посилання на texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf, and texindex

Встановлені бібліотеки: MiscXS.so, ParseTeXi.so, XSPParagraph.so (все в */usr/lib/texinfo*)

Встановлені директорії: */usr/share/texinfo* and */usr/lib/texinfo*

Короткі описи

info

Використовується для читання інформаційних сторінок, які схожі на сторінки **man**, але часто містять набагато більше інформації, ніж просто пояснення всіх доступних опцій командного рядка [Наприклад, порівняйте **man bison** та **info bison**.]

install-info

Використовується для встановлення інформаційних сторінок; оновлює записи в індексному файлі **info**.

makeinfo

Перекладає задані вихідні документи Texinfo в інформаційні сторінки, звичайний текст або HTML.

pdftexi2dvi

Використовується для форматування заданого документа Texinfo у файл Portable Document Format (PDF).

pod2texi

Конвертує Pod у формат Texinfo.

texi2any

Перекладає вихідну документацію Texinfo в різні інші формати.

texi2dvi

Використовується для форматування заданого документа Texinfo у файл, незалежний від пристрою, який можна надрукувати.

texi2pdf

Використовується для форматування заданого документа Texinfo у файл Portable Document Format (PDF).

texindex

Використовується для сортування індексних файлів Texinfo.

8.73. Vim-9.1.1166

Пакет Vim містить потужний текстовий редактор.

Приблизний час побудови: 3.4 SBU

Необхідний простір на диску: 251 MB



Альтернативи Vim

Якщо ви віддаєте перевагу іншому редактору, такому як Emacs, Joe або Nano, перегляньте <https://www.linuxfromscratch.org/blfs/view/12.3/postlfs/editors.html>, де ви знайдете рекомендації щодо встановлення.

8.73.1. Інсталяція Vim

Спочатку змініть місце розташування файлу конфігурації vimrc за замовчуванням на /etc:

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Підготуйте Vim до компіляції:

```
./configure --prefix=/usr
```

Скомпілюйте пакет:

```
make
```

Щоб підготувати тести, переконайтесь, що користувач `tester` може записувати в дерево джерел і виключіть один файл, що містить тести, які вимагають `curl` або `wget`:

```
chown -R tester .
sed '/test_plugin_glvs/d' -i src/testdir/Make_all.mak
```

Тепер запустіть тести як користувач `tester`:

```
su tester -c "TERM=xterm-256color LANG=en_US.UTF-8 make -j1 test" \
&> vim-test.log
```

Набір тестів виводить на екран велику кількість бінарних даних. Це може спричинити проблеми з налаштуваннями поточного терміналу (особливо коли ми перезаписуємо змінну `TERM`, щоб задоволінити деякі припущення набору тестів). Цю проблему можна уникнути, перенаправивши вивід у файл журналу, як показано вище. У разі успішного виконання тесту в файлі журналу з'явиться напис ALL DONE.

Встановіть пакет:

```
make install
```

Багато користувачів рефлекторно вводять `vi` замість `vim`. Щоб дозволити виконання `vim`, коли користувачі звичнно вводять `vi`, створіть символічне посилання як для бінарного файлу, так і для сторінки тап на наданих мовах:

```
ln -sv vim /usr/bin/vi
for L in /usr/share/man/{,*/}man1/vim.1; do
    ln -sv vim.1 $(dirname $L)/vi.1
done
```

За замовчуванням документація Vim встановлюється в `/usr/share/vim`. Наступне символічне посилання дозволяє отримати доступ до документації через `/usr/share/doc/vim-9.1.1166`, що робить її сумісною з розташуванням документації для інших пакетів:

```
ln -sv ../../vim/vim91/doc /usr/share/doc/vim-9.1.1166
```

Якщо на систему LFS буде встановлено X Window System, після встановлення X може знадобитися перекомпіляція Vim. Vim постачається з графічною версією редактора, яка вимагає встановлення X та деяких додаткових бібліотек. Для отримання додаткової інформації про цей процес зверніться до документації Vim та сторінки встановлення Vim у книзі BLFS за адресою <https://www.linuxfromscratch.org/blfs/view/12.3/postlfs/vim.html>.

8.73.2. Конфігурація Vim

За замовчуванням **vim** працює в режимі, несумісному з **vi**. Це може бути новим для користувачів, які раніше використовували інші редактори. Налаштування «*poscompatible*» включено нижче, щоб підкреслити факт використання нового режиму роботи. Воно також нагадує тим, хто хоче перейти в режим «*compatible*», що це має бути першим налаштуванням у файлі конфігурації. Це необхідно, оскільки воно змінює інші налаштування, і заміщення повинні йти після цього налаштування. Створіть файл конфігурації **vim** за замовчуванням, виконавши наступне:

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc

" Ensure defaults are set before customizing settings, not after
source $VIMRUNTIME/defaults.vim
let skip_defaults_vim=1

set nocompatible
set backspace=2
set mouse=
syntax on
if (&term == "xterm") || (&term == "putty")
    set background=dark
endif

" End /etc/vimrc
EOF
```

Налаштування `set nocompatible` змушує **vim** поводитися більш корисним чином (за замовчуванням), ніж у режимі сумісності з **vi**. Видаліть «*по*», щоб зберегти стару поведінку **vi**. Налаштування `set backspace=2` дозволяє видаляти символи за межами рядків, автоматичних відступів та початку вставки. Параметр `syntax on` вмикає підсвічування синтаксису **vim**. Налаштування `set mouse=` увімкне правильне вставлення тексту за допомогою миші під час роботи в `chroot` або через віддалене з'єднання. Нарешті, оператор `if` з налаштуванням `set background=dark` виправляє припущення **vim** щодо кольору фону деяких емуляторів терміналів. Це надає підсвічуванню кращу колірну схему для використання на чорному фоні цих програм.

Документацію щодо інших доступних опцій можна отримати, виконавши таку команду:

```
vim -c ':options'
```

**Note**

За замовчуванням, **vim** встановлює лише файли перевірки орфографії для англійської мови. Щоб встановити файли перевірки орфографії для вашої улюбленої мови, скопіюйте файли `.spl` і, за бажанням, файли `.sug` для вашої мови та кодування символів з `runtime/spell` до `/usr/share/vim/vim91/spell/`.

Щоб використовувати ці файли перевірки орфографії, потрібно виконати деякі налаштування в `/etc/vimrc`, наприклад:

```
set spelllang=en,ru
set spell
```

Для отримання додаткової інформації див. `runtime/spell/README.txt`.

8.73.3. Вміст Vim

Встановлені програми:	ex (посилання на vim), rview (посилання на vim), rvim (посилання на vim), vi (посилання на vim), view (посилання на vim), vim, vimdiff (посилання на vim), vimtutor, xxd
Встановлені директорії:	/usr/share/vim

Короткі описи

ex	Запускає vim в режимі ex.
rview	Є обмеженою версією view ; не можна запускати команди оболонки та призупиняти view .
rvim	Є обмеженою версією vim ; не можна запускати команди оболонки та призупиняти vim .
vi	Посилання на vim .
view	Запускає vim у режимі тільки для читання.
vim	Є редактором.
vimdiff	Редагує дві або три версії файлу за допомогою vim і показує відмінності.
vimtutor	Навчає основним клавішам і командам vim .
xxd	Створює шістнадцятковий дамп заданого файлу; також може виконувати зворотну операцію, тому може використовуватися для бінарного патчування.

8.74. MarkupSafe-3.0.2

MarkupSafe — це модуль Python, який реалізує безпечний для розмітки XML/HTML/XHTML рядок.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 500 KB

8.74.1. Інсталяція MarkupSafe

Зкомпілюйте MarkupSafe згідно наступної команди:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Цей пакет не містить набору тестів.

Встановіть пакет:

```
pip3 install --no-index --find-links dist MarkupSafe
```

8.74.2. Вміст MarkupSafe

Встановлені директорії: /usr/lib/python3.13/site-packages/MarkupSafe-3.0.2.dist-info

8.75. Jinja2-3.1.5

Jinja2 — це модуль Python, який реалізує просту мову шаблонів Python.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 2.5 MB

8.75.1. Інсталяція Jinja2

Зберіть пакет:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Встановіть пакет:

```
pip3 install --no-index --find-links dist Jinja2
```

8.75.2. Вміст Jinja2

Встановлені директорії: /usr/lib/python3.13/site-packages/Jinja2-3.1.5.dist-info

8.76. Udev із Systemd-257.3

Пакет Udev містить програми для динамічного створення вузлів пристройів.

Приблизний час побудови: 0.3 SBU

Необхідний простір на диску: 161 MB

8.76.1. Інсталляція Udev

Udev є частиною пакета systemd-257.3. Використовуйте файл `systemd-257.3.tar.xz` як вихідний архів.

Видаліть дві непотрібні групи `render` та `sgx`, із правил udev за замовчуванням:

```
sed -e 's/GROUP="render"/GROUP="video"/' \
    -e 's/GROUP="sgx", //' \
    -i rules.d/50-udev-default.rules.in
```

Видаліть одне правило udev, яке вимагає повної інсталляції Systemd:

```
sed -i '/systemd-sysctl/s/^#/!' rules.d/99-systemd.rules.in
```

Налаштуйте жорстко задані шляхи до файлів конфігурації мережі для автономної інсталляції udev:

```
sed -e '/NETWORK_DIRS/s/systemd/udev/' \
    -i src/libsystemd/sd-network/network-util.h
```

Підготуйте Udev до компіляції:

```
mkdir -p build
cd      build

meson setup .. \
    --prefix=/usr \
    --buildtype=release \
    -D mode=release \
    -D dev-kvm-mode=0660 \
    -D link-udev-shared=false \
    -D logind=false \
    -D vconsole=false
```

Значення параметрів meson:

`--buildtype=release`

Цей перемикач замінює типовий тип збірки («`debug`»), який створює неоптимізовані бінарні файли.

`-D mode=release`

Вимикає деякі функції, які вважаються експериментальними.

`-D dev-kvm-mode=0660`

За замовчуванням правило udev дозволяє всім користувачам отримати доступ до `/dev/kvm`.

Редактори вважають це небезпечним. Ця опція перекриває це правило.

`-D link-udev-shared=false`

Ця опція запобігає зв'язку udev із внутрішньою динамічною бібліотекою `systemd`, `libsystemd-shared`.

Ця бібліотека призначена для спільного використання багатьма компонентами `Systemd` і є надто потужною для інсталляції, що використовує лише udev.

```
-D logind=false -D vconsole=false
```

Ці опції запобігають створенню декількох файлів правил udev, що належать до інших компонентів Systemd, які ми не будемо встановлювати.

Отримайте список відправлених допоміжних програм udev і збережіть його в змінній середовища (експорт не є обов'язковим, але це спрошує збірку як звичайному користувачеві або при використанні менеджера пакетів):

```
export udev_helpers=$(grep "'name' :" ./src/udev/meson.build | \
    awk '{print $3}' | tr -d "," | grep -v 'udevadm')
```

Створіть лише компоненти, необхідні для udev:

```
ninja udevadm systemd-hwdb \
    $(ninja -n | grep -Eo '(src/(lib)?udev|rules.d|hwdb.d)/[^ ]*') \
    $(realpath libudev.so --relative-to .) \
    $udev_helpers
```

Встановіть пакет:

```
install -vm755 -d {/usr/lib,/etc}/udev/{hwdb.d, rules.d, network}
install -vm755 -d /usr/{lib,share}/pkgconfig
install -vm755 udevadm                               /usr/bin/
install -vm755 systemd-hwdb                         /usr/bin/udev-hwdb
ln      -svfn ../bin/udevadm                         /usr/sbin/udevd
cp      -av libudev.so{,*[0-9]}                      /usr/lib/
install -vm644 ../src/libudev/libudev.h              /usr/include/
install -vm644 src/libudev/*.pc                      /usr/lib/pkgconfig/
install -vm644 src/udev/*.pc                         /usr/share/pkgconfig/
install -vm644 ../src/udev/udev.conf                /etc/udev/
install -vm644 rules.d/* ../rules.d/README          /usr/lib/udev/rules.d/
install -vm644 $(find ../rules.d/*.rules \
                  -not -name '*power-switch*') /usr/lib/udev/rules.d/
install -vm644 hwdb.d/* ../hwdb.d/{*.hwdb, README} /usr/lib/udev/hwdb.d/
install -vm755 $udev_helpers                          /usr/lib/udev
install -vm644 ../network/99-default.link           /usr/lib/udev/network
```

Встановіть деякі власні правила та файли підтримки, корисні в середовищі LFS:

```
tar -xvf ../../udev-lfs-20230818.tar.xz
make -f udev-lfs-20230818/Makefile.lfs install
```

Встановіть сторінки man:

```

tar -xf ../../systemd-man-pages-257.3.tar.xz \
--no-same-owner --strip-components=1 \
-C /usr/share/man --wildcards '*/*udev*' '*/*libudev*' \
'*/*systemd.link.5' \
'*/*systemd-{hwdb,udevd.service}.8

sed 's|systemd/network|udev/network|' \
/usr/share/man/man5/systemd.link.5 \
> /usr/share/man/man5/udev.link.5

sed 's/systemd\\(\\\\?-\\\)/udev\\1/' /usr/share/man/man8/systemd-hwdb.8 \
> /usr/share/man/man8/udev-hwdb.8

sed 's|lib.*udevd|sbin/udevd|' \
/usr/share/man/man8/systemd-udevd.service.8 \
> /usr/share/man/man8/udevd.8

rm /usr/share/man/man*/systemd*

```

На сам кінець, зніміть змінну `udev_helpers`:

```
unset udev_helpers
```

8.76.2. Конфігурація Udev

Інформація про апаратні пристрої зберігається в каталогах `/etc/udev/hwdb.d` та `/usr/lib/udev/hwdb.d`. Udev потребує, щоб ця інформація була скомпільована в бінарну базу даних `/etc/udev/hwdb.bin`. Створіть початкову базу даних:

```
udev-hwdb update
```

Цю команду потрібно виконувати кожного разу, коли оновлюється інформація про апаратне забезпечення.

8.76.3. Вміст Udev

Встановлені програми:	<code>udevadm</code> , <code>udevd</code> (символічне посилання на <code>udevadm</code>), <code>udev-hwdb</code>
Встановлені бібліотеки:	<code>libudev.so</code>
Встановлені директорії:	<code>/etc/udev</code> та <code>/usr/lib/udev</code>

Короткі описи

udevadm	Загальний інструмент адміністрування udev: керує демоном <code>udevd</code> , надає інформацію з бази даних Udev, відстежує <code>uevents</code> , очікує завершення <code>uevents</code> , перевіряє конфігурацію Udev та запускає <code>uevents</code> для певного пристроя
udevd	Демон, який слухає <code>uevents</code> на сокеті <code>netlink</code> , створює пристрой та запускає налаштовані зовнішні програми у відповідь на ці <code>uevents</code>
udev-hwdb	Оновлює або запитує базу даних обладнання.

libudev

Бібліотечний інтерфейс до інформації про пристрої udev

/etc/udev

Містить файли конфігурації Udev, дозволи для пристроїв та правила для іменування пристроїв

8.77. Man-DB-2.13.0

Пакет Man-DB містить програми для пошуку та перегляду сторінок довідки.

Приблизний час побудови: 0.3 SBU

Необхідний простір на диску: 44 MB

8.77.1. Інсталяція Man-DB

Підготуйте Man-DB до компіляції:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/man-db-2.13.0 \
            --sysconfdir=/etc \
            --disable-setuid \
            --enable-cache-owner=bin \
            --with-browser=/usr/bin/lynx \
            --with-vgrind=/usr/bin/vgrind \
            --with-grap=/usr/bin/grap \
            --with-systemdtmpfilesdir= \
            --with-systemdsystemunitdir=
```

Значення конфігураційних параметрів:

--disable-setuid

Це вимикає встановлення програми **man** setuid для користувача **man**.

--enable-cache-owner=bin

Це змінює власника системних файлів кешу на користувача **bin**.

--with-...

Ці три параметри використовуються для встановлення деяких програм за замовчуванням. **lynx** — це текстовий веб-браузер (див. BLFS для інструкцій з встановлення), **vgrind** перетворює вихідні коди програм у вхідні дані Groff, а **grap** корисний для верстання графіків у документах Groff. Програми **vgrind** та **grap** зазвичай не потрібні для перегляду сторінок довідки. Вони не входять до складу LFS або BLFS, але ви можете встановити їх самостійно після завершення LFS, якщо бажаєте.

--with-systemd...

Ці параметри запобігають встановленню непотрібних каталогів та файлів systemd.

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Встановіть пакет:

```
make install
```

8.77.2. Не англійські сторінки довідки в LFS

У наступній таблиці наведено набір символів, який Man-DB припускає, що сторінки довідки, встановлені в `/usr/share/man/<11>`, будуть закодовані. Крім того, Man-DB правильно визначає, чи сторінки довідки, встановлені в цьому каталозі, закодовані в UTF-8.

Таблиця 8.1. Очікуване кодування символів у старих 8-бітних сторінках довідки

Мова(код)	Кодування	Мова(код)	Кодування
Данська (da)	ISO-8859-1	Хорватська (hr)	ISO-8859-2
Німецька (de)	ISO-8859-1	Угорська (hu)	ISO-8859-2
Англійська (en)	ISO-8859-1	Японська (ja)	EUC-JP
Іспанська (es)	ISO-8859-1	Корейська (ko)	EUC-KR
Естонська (et)	ISO-8859-1	Литовська (lt)	ISO-8859-13
Фінська (fi)	ISO-8859-1	Латиська (lv)	ISO-8859-13
Французька (fr)	ISO-8859-1	Македонська (mk)	ISO-8859-5
Ірландська (ga)	ISO-8859-1	Польська (pl)	ISO-8859-2
Галісійська (gl)	ISO-8859-1	Румунська (ro)	ISO-8859-2
Індонезійська (id)	ISO-8859-1	Грецька (el)	ISO-8859-7
Ісландська (is)	ISO-8859-1	Словачька (sk)	ISO-8859-2
Італійська (it)	ISO-8859-1	Словенська (sl)	ISO-8859-2
Норвезька Bokmal (nb)	ISO-8859-1	Сербська Latin (sr@latin)	ISO-8859-2
Данська (nl)	ISO-8859-1	Сербська (sr)	ISO-8859-5
Норвезька Nynorsk (nn)	ISO-8859-1	Турецька (tr)	ISO-8859-9
Норвезька (no)	ISO-8859-1	Українська (uk)	KOI8-U
Португальська (pt)	ISO-8859-1	В'єтнамська (vi)	TCVN5712-1
Шведська (sv)	ISO-8859-1	Спрощена Китайська (zh_CN)	GBK
Білоруська (be)	CP1251	Традиційна Китайська, Сінгапур (zh_SG)	GBK
Болгарська (bg)	CP1251	Традиційна Китайська, Гонконг (zh_HK)	BIG5HKSCS
Чеська (cs)	ISO-8859-2	Традиційна Китайська (zh_TW)	BIG5



Note

Сторінки довідки мовами, що не входять до списку, не підтримуються.

8.77.3. Вміст Man-DB

Встановлені програми:	accessdb, apropos (посилання на whatis), catman, lexgrog, man, man-recode, mandb, manpath, whatis
Встановлені бібліотеки:	libman.so та libmandb.so (both in /usr/lib/man-db)
Встановлені директорії:	/usr/lib/man-db, /usr/libexec/man-db, /usr/share/doc/man-db-2.13.0

Короткі описи

accessdb	Виводить вміст бази даних whatis у вигляді, зрозумілому для людини.
apropos	Шукає в базі даних whatis і відображає короткі описи системних команд, що містять заданий рядок
catman	Створює або оновлює попередньо відформатовані сторінки довідки
lexgrog	Відображає однорядкову зведену інформацію про задану сторінку довідки
man	Форматує і відображає запитувану сторінку довідки
man-recode	Конвертує сторінки довідки в інше кодування
mandb	Створює або оновлює базу даних whatis
manpath	Відображає вміст \$MANPATH або (якщо \$MANPATH не встановлено) відповідний шлях пошуку на основі налаштувань у man.conf та середовища користувача
whatis	Шукає в базі даних whatis і відображає короткі описи системних команд, що містять задане ключове слово як окреме слово
libman	Містить підтримку виконання для man
	Містить підтримку виконання для man

8.78. Procps-ng-4.0.5

Пакет Procps-ng містить програми для моніторингу процесів.

Приблизний час побудови: 0.1 SBU

Необхідний простір на диску: 28 MB

8.78.1. Інсталяція Procps-ng

Підготуйте Procps-ng до компіляції:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/procps-ng-4.0.5 \
            --disable-static \
            --disable-kill \
            --enable-watch8bit \
```

Значення конфігураційних параметрів:

--disable-kill

Цей перемикач вимикає створення команди **kill**; вона буде встановлена з пакета Util-linux.

--enable-watch8bit

Цей перемикач вимикає підтримку ncursesw для команди **watch**, щоб вона могла обробляти 8-бітні символи.

Скомпілюйте пакет:

```
make
```

Запустіть тестовий набір, виконайте:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Відомо, що один тест під назвою **ps** with output flag bsdtime, cputime, etime, etimes не працює, якщо ядро хоста не побудовано з увімкненим CONFIG_BSD_PROCESS_ACCT. Крім того, один тест pgrep може не працювати в середовищі chroot.

Встановіть пакет:

```
make install
```

8.78.2. Вміст Procps-ng

Встановлені програми: free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w, watch

Встановлені бібліотеки: libproc-2.so

Встановлені директорії: /usr/include/procps та /usr/share/doc/procps-ng-4.0.5

Короткі описи

free

Повідомляє про обсяг вільної та використаної пам'яті (як фізичної, так і підкачки) в системі

pgrep	Шукає процеси за їхнім іменем та іншими атриутами
pidof	Повідомляє про PID заданих програм
pkill	Посилає сигнали процесам за їхнім іменем та іншими атрибутами
pmap	Повідомляє про карту пам'яті заданого процесу
ps	Перелічує поточні запущені процеси.
pwdx	Повідомляє про поточний робочий каталог процесу.
slabtop	Відображає детальну інформацію про кеш ядра в режимі реального часу.
sysctl	Змінює параметри ядра під час роботи.
tload	Виводить графік поточного середнього навантаження системи.
top	Відображає список процесів, що найбільше навантажують процесор; забезпечує постійний огляд активності процесора в режимі реального часу.
uptime	Повідомляє, скільки часу система працює, скільки користувачів увійшли в систему та середнє навантаження системи
vmstat	Повідомляє статистику віртуальної пам'яті, надаючи інформацію про процеси, пам'ять, підкачку, блоковий ввід/ вивід (IO), переривання та активність процесора
w	Показує, які користувачі зараз увійшли в систему, де і з якого часу
watch	Виконує задану команду повторно, відображаючи перший екран її виводу; це дозволяє користувачеві спостерігати за зміною виводу з часом
libproc-2	Містить функції, які використовуються більшістю програм у цьому пакеті

8.79. Util-linux-2.40.4

Пакет Util-linux містить різні утиліти. Серед них є утиліти для роботи з файловими системами, консолями, розділами та повідомленнями.

Приблизний час побудови: 0.5 SBU

Необхідний простір на диску: 316 MB

8.79.1. Інсталяція Util-linux

Підготуйте Util-linux до компіляції:

```
./configure --bindir=/usr/bin \
            --libdir=/usr/lib \
            --runstatedir=/run \
            --sbindir=/usr/sbin \
            --disable-chfn-chsh \
            --disable-login \
            --disable-nologin \
            --disable-su \
            --disable-setpriv \
            --disable-runuser \
            --disable-pylibmount \
            --disable-liblastlog2 \
            --disable-static \
            --without-python \
            --without-systemd \
            --without-systemdsystemunitdir \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.40.4
```

Параметри `--disable` та `--without` запобігають появи попереджень про складання компонентів, які вимагають пакетів, що не входять до складу LFS, або несумісні з програмами, встановленими іншими пакетами.

Скомпілюйте пакет:

```
make
```

За бажанням, створіть фіктивний файл `/etc/fstab`, щоб задовольнити дві умови тестування, і запустіть набір тестів як користувач, що не є `root`:



Warning

Виконання набору тестів як користувач `root` може бути шкідливим для вашої системи. Для його виконання опція `CONFIG_SCSI_DEBUG` для ядра повинна бути доступною в поточній системі і побудована як модуль. Вбудування її в ядро завадить завантаженню. Для повного охоплення необхідно встановити інші пакети BLFS. Якщо бажаєте, цей тест можна виконати, завантажившись у готову систему LFS і виконавши:

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

```
touch /etc/fstab
chown -R tester .
su tester -c "make -k check"
```

Тести *hardlink* завершаться невдачею, якщо в ядрі хоста не ввімкнено опцію `CONFIG_CRYPTO_USER_API_HASH` або не ввімкнено жодної опції, що забезпечує реалізацію SHA256 (наприклад, `CONFIG_CRYPTO_SHA256` або `CONFIG_CRYPTO_SHA256_SSSE3`, якщо процесор підтримує Supplemental SSE3). Крім того, тест *lsfd: inotify* завершиться невдачею, якщо в ядрі не ввімкнено опцію `CONFIG_NETLINK_DIAG`.

Два інших тести, *lsfd: SOURCE* column та *utmp: last*, як відомо, не працюють у середовищі chroot.

Встановіть пакет:

```
make install
```

8.79.2. Вміст Util-linux

Встановлені програми:

addpart, agetty, blkdiscard, blkid, blkzone, blockdev, cal, cfdisk, chcpu, chmem, choom, chrt, col, colcrt, colrm, column, ctrlaltdel, delpart, dmesg, eject, falllocate, fdisk, fincore, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hardlink, hexdump, hwclock, i386 (посилання на setarch), ionice, ipcmk, ipcrm, ipcs, irqtop, isosize, kill, last, lastb (link to last), lddattach, linux32 (посилання на setarch), linux64 (link to setarch), logger, look, losetup, lsblk, lscpu, lsipc, lsirq, lsfd, lslocks, lslogins, lsmem, lsns, mcookie, mesg, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, nsenter, partx, pivot_root, prlimit, readprofile, rename, renice, resizepart, rev, rfkill, rtcwake, script, scriptlive, scriptreplay, setarch, setsid, setterm, sfdisk, sudo, swaplabel, swapoff, swapon, switch_root, taskset, uclampset, ul, umount, uname26 (link to setarch), unshare, utmpdump, uid, uuidgen, uuidparse, wall, wdctl, whereis, wipefs, x86_64 (link to setarch), zramctl

Встановлені бібліотеки:

libblkid.so, libfdisk.so, libmount.so, libsmartcols.so, libuuid.so

Встановлені директорії:

/usr/include/blkid, /usr/include/libfdisk, /usr/include/libmount, /usr/include/libsmartcols, /usr/include/uuid, /usr/share/doc/util-linux-2.40.4, /var/lib/hwclock

Короткі описи

addpart

Інформує ядро Linux про нові розділи.

agetty

Відкриває порт tty, запитує ім'я для входу, а потім запускає програму **login**.

blkdiscard

Видаляє сектори на пристрой.

blkid

Командна утиліта для пошуку та виведення атрибутів блокових пристройів.

blkzone

Використовується для управління зонуваними блоковими пристроями зберігання даних.

blockdev	Дозволяє користувачам викликати ioctl's блокових пристрів з командного рядка
cal	Відображає простий календар
fdisk	Маніпулює таблицею розділів заданого пристрою
chcpu	Змінює стан процесорів
chmem	Налаштовує пам'ять
choom	Відображає та налаштовує бали OOM-killer, які використовуються для визначення, який процес слід вбити першим, коли Linux не має достатньо пам'яті
chrt	Маніпулює атрибутами процесу в режимі реального часу.
col	Відфільтровує зворотні переноси рядків.
colcrt	Відфільтровує вихідні дані nroff для терміналів, які не мають деяких можливостей, таких як перезапис і напіврядки.
colrm	Відфільтровує задані стовпці.
column	Форматує заданий файл у кілька стовпців.
ctrlaltdel	Встановлює функцію комбінації клавіш Ctrl+Alt+Del для жорсткого або м'якого перезавантаження.
delpart	Запитує ядро Linux про видалення розділу.
dmesg	Виводить повідомлення ядра про завантаження.
eject	Видає знімний носій.
fallocate	Попередньо виділяє місце для файлу.
fdisk	Маніпулює таблицею розділів заданого пристрою.
fincore	Підраховує сторінки вмісту файлу в ядрі.
findfs	Знаходить файлову систему за міткою або універсальним унікальним ідентифікатором (UUID).
findmnt	Є інтерфейсом командного рядка до бібліотеки libmount для роботи з файлами mountinfo, fstab і mtab.
flock	Отримує блокування файлу, а потім виконує команду з утриманням блокування
fsck	Використовується для перевірки та, за бажанням, відновлення файлових систем
fsck.cramfs	Виконує перевірку цілісності файлової системи Cramfs на заданому пристрої
fsck.minix	Виконує перевірку цілісності файлової системи Minix на заданому пристрої
fsfreeze	Є дуже простим обгорткою навколо операцій драйвера ядра

	FIFREEZE/FITHAW ioctl
fstrim	Відкидає невикористані блоки на змонтованій файловій системі
getopt	Аналізує опції в заданому командному рядку
hardlink	Об'єднує дублікати файлів, створюючи жорсткі посилання
hexdump	Виводить заданий файл у шістнадцятковому, десятковому, вісімковому або ASCII-форматі
hwclock	Зчитує або встановлює системний апаратний годинник, також званий годинником реального часу (RTC) або годинником базової системи вводу-виводу (BIOS)
i386	Символічне посилання на setarch
ionice	Отримує або встановлює клас планування вводу-виводу та пріоритет для програми
ipcmk	Створює різні ресурси IPC
ipcrm	Видаляє вказаний ресурс міжпроцесної комунікації (IPC)
ipcs	Надає інформацію про стан IPC
irqtop	Відображає інформацію про лічильник переривань ядра у вигляді top(1)
isosize	Повідомляє розмір файлової системи iso9660
kill	Відправляє сигнали процесам
last	Показує, які користувачі останнім часом входили (і виходили) в систему, здійснюючи пошук у файлі /var/log/wtmp; також показує завантаження системи, вимкнення та зміни рівня запуску
lastb	Показує невдалі спроби входу, як записано в /var/log/btmp
lattach	Приєднує дисципліну лінії до послідовної лінії
linux32	Символічне посилання на setarch
linux64	Символічне посилання на setarch
logger	Вводить вказане повідомлення в системний журнал
look	Відображає рядки, що починаються з вказаного рядка
losetup	Налаштовує та контролює петлеві пристрої
lsblk	Виводить інформацію про всі або вибрані блокові пристрої у вигляді дерева.
lscpu	Виводить інформацію про архітектуру процесора.
lsfd	Виводить інформацію про відкриті файли; замінює lsof .
lipc	Виводить інформацію про засоби IPC, які зараз використовуються в системі.
lsirq	Виводить інформацію про лічильник переривань ядра.

lslocks	Виводить список локальних блокувань системи.
lslogins	Виводить інформацію про користувачів, групи та системні облікові записи.
lsmem	Виводить діапазони доступної пам'яті з їхнім статусом онлайн.
lsns	Перелічує простори імен.
mcookie	Генерує магічні куки (128-бітні випадкові шістнадцяткові числа) для xauth .
mesg	Контролює, чи можуть інші користувачі надсилати повідомлення на термінал поточного користувача.
mkfs	Створює файлову систему на пристрої (зазвичай на розділі жорсткого диска).
mkfs.bfs	Створює файлову систему Santa Cruz Operations (SCO) bfs.
mkfs.cramfs	Створює файлову систему cramfs.
mkfs.minix	Створює файлову систему Minix.
mkswap	Ініціалізує вказаний пристрій або файл для використання як область підкачки
more	Фільтр для перегляду тексту по одному екрану за раз
mount	Приєднує файлову систему на вказаному пристрої до вказаного каталогу в дереві файлової системи
mountpoint	Перевіряє, чи є каталог точкою монтування
namei	Показує символічні посилання у вказаних шляхах
nsenter	Запускає програму з просторами імен інших процесів
partx	Повідомляє ядро про наявність і нумерацію розділів на диску
pivot_root	Робить вказану файлову систему новою кореневою файловою системою поточного процесу.
prlimit	Отримує та встановлює обмеження ресурсів процесу.
readprofile	Зчитує інформацію про профілювання ядра.
rename	Перейменовує вказані файли, замінюючи вказаний рядок іншим.
renice	Змінює пріоритет запущених процесів.
resizepart	Запитує ядро Linux про зміну розміру розділу.
rev	Змінює порядок рядків у вказаному файлі.
rfkill	Інструмент для ввімкнення та вимкнення бездротових пристрійів.
rtcwake	Використовується для переведення системи в режим сну до вказаного часу пробудження.
script	Створює стенограму сеансу терміналу.

scriptlive	Повторно запускає стенограми сеансів, використовуючи інформацію про час
scriptreplay	Відтворює стенограми, використовуючи інформацію про час
setarch	Змінює заявлену архітектуру в новому програмному середовищі та встановлює прапорці особистості
setsid	Запускає вказану програму в новому сеанси
setterm	Встановлює атрибути терміналу.
sfdisk	Маніпулятор таблиці розділів диска.
sulogin	Дозволяє root входити в систему; зазвичай викликається init , коли система переходить в режим одного користувача.
swaplabel	Вносить зміни в UUID і мітку області підкачки.
swapoff	Вимикає пристрій та файли для підкачки та обміну.
swapon	Вмикає пристрій та файли для підкачки та обміну, а також перелічує пристрій та файли, які зараз використовуються.
switch_root	Перемикається на іншу файлову систему як корінь дерева монтування.
taskset	Отримує або встановлює афінність процесора до процесу.
uclampset	Маніпулює атрибутами обмеження використання системи або процесу.
ul	Фільтр для перетворення підкреслень в екранні послідовності, що вказують на підкреслення для термінала, що використовується
umount	Відключає файлову систему від дерева файлів системи
uname26	Символічне посилання на setarch
unshare	Запускає програму з деякими просторами імен, не спільними з батьківським
utmpdump	Відображає вміст заданого файлу входу в зручному для користувача форматі
uuidd	Демон, який використовується бібліотекою UUID для генерації UUID на основі часу безпечним та гарантовано унікальним способом
uuidgen	Створює нові UUID. Кожен новий UUID є випадковим числом, яке, ймовірно, буде унікальним серед усіх UUID, створених у локальній системі та в інших системах, у минулому та в майбутньому, з надзвичайно високою ймовірністю (можливі 2 UUID)
uuidparse	Утиліта для аналізу унікальних ідентифікаторів
wall	Відображає вміст файла або, за замовчуванням, його стандартний вхід, на терміналах усіх користувачів, які наразі зареєстровані
wdctl	Показує стан апаратного сторожового таймера
whereis	Повідомляє про розташування бінарних, вихідних та сторінок довідки

для заданої команди

wipefs

Видаляє підпис файлової системи з пристрою

x86_64

Символічне посилання на setarch

zramctl

Програма для налаштування та керування пристроями zram (стиснений диск оперативної пам'яті)

libblkid

Містить процедури для ідентифікації пристройів та вилучення токенів

libfdisk

Містить процедури для маніпулювання таблицями розділів

libmount

Містить процедури для монтування та демонтування блокових пристройів

libsmartcols

Містить процедури для сприяння виведенню на екран у табличній формі

libuuid

Містить процедури для генерації унікальних ідентифікаторів для об'єктів, які можуть бути доступними за межами локальної системи

8.80. E2fsprogs-1.47.2

Пакет E2fsprogs містить утиліти для роботи з файловою системою ext2. Він також підтримує файлові системи ext3 та ext4 з журналуванням.

Приблизний час побудови: 2.4 SBU на HDD, 0.5 SBU на SSD

Необхідний простір на диску: 99 MB

8.80.1. Інсталяція E2fsprogs

Документація E2fsprogs рекомендує компілювати пакет у підкаталозі дерева джерел:

```
mkdir -v build
cd      build
```

Підготуйте E2fsprogs до компіляції:

```
./configure --prefix=/usr \
            --sysconfdir=/etc \
            --enable-elf-shlibs \
            --disable-libblkid \
            --disable-libuuid \
            --disable-uuid \
            --disable-fsck
```

Значення конфігураційних параметрів:

--enable-elf-shlibs

Це створює спільні бібліотеки, які використовують деякі програми в цьому пакеті.

--disable-*

Це запобігає створенню та встановленню бібліотек libuuid та libblkid, демона uuidd та обортки **fsck**; util-linux встановлює більш нові версії.

Скомпілюйте пакет:

```
make
```

Щоб перевірити результати, виконайте:

```
make check
```

Відомо, що один тест під назвою `m_assume_storage_prezeroed` не проходить. Інший тест під назвою `m_rootdir_acl` не проходить, якщо файлова система, яка використовується для системи LFS, не є ext4.

Встановіть пакет:

```
make install
```

Видаліть непотрібні статичні бібліотеки:

```
rm -fv /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Цей пакет встановлює gzip-файл `.info`, але не оновлює системний файл `dir`. Розпакуйте цей файл, а потім оновлюйте системний файл `dir` за допомогою таких команд:

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir /usr/share/info/libext2fs.info
```

За бажанням, створіть та встановіть додаткову документацію, виконавши наступні команди:

```
makeinfo -o doc/com_err.info .. /lib /et /com_err.texinfo
install -v -m644 doc/com_err.info /usr/share/info
install-info --dir-file=/usr/share/info/dir /usr/share/info/com_err.info
```

8.80.2. Конфігурація E2fsprogs

/etc/mke2fs.conf містить значення за замовчуванням різних опцій командного рядка **mke2fs**. Ви можете редагувати цей файл, щоб значення за замовчуванням відповідали вашим потребам. Наприклад, деякі утиліти (не в LFS або BLFS) не можуть розпізнати файлову систему ext4 з увімкненою функцією metadata_csum_seed. Якщо вам потрібна така утиліта, ви можете видалити цю функцію зі списку функцій ext4 за замовчуванням за допомогою команди:

```
sed 's/metadata_csum_seed, //' -i /etc/mke2fs.conf
```

Детальнішу інформацію дивіться у довідковій сторінці *mke2fs.conf(5)*.

8.80.3. Вміст E2fsprogs

Встановлені програми:	badblocks, chattr, compile_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2label, e2mmpstatus, e2scrub, e2scrub_all, e2undo, e4crypt, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mklost+found, resize2fs, tune2fs
Встановлені бібліотеки:	libcom_err.so, libe2p.so, libext2fs.so, libss.so
Встановлені директорії:	/usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/ss, /usr/lib/e2fsprogs, /usr/share/et, /usr/share/ss

Короткі описи

badblocks	Шукає на пристрої (зазвичай на розділі диска) пошкоджені блоки
chattr	Змінює атрибути файлів у файлових системах ext{2,3,4}
compile_et	Компілятор таблиць помилок; перетворює таблицю імен кодів помилок та повідомлень у вихідний файл C придатний для використання з бібліотекою com_err
debugfs	Відладчик файлової системи; може використовуватися для перевірки та зміни стану файлових систем ext{2,3,4}
dumpe2fs	Виводить інформацію про суперблок та групи блоків для файлової системи, що знаходиться на заданому пристрої
e2freefrag	Повідомляє інформацію про фрагментацію вільного простору
e2fsck	Використовується для перевірки та, за бажанням, відновлення файлових систем ext{2,3,4}
e2image	Використовується для збереження критичних даних файлової системи ext{2,3,4} у файлі
e2label	Відображає або змінює мітку файлової системи на файловій системі

e2mmpstatus	Перевіряє стан MMP (Multiple Mount Protection) файлової системи ext{2,3,4}
e2scrub	Перевіряє вміст змонтованої файлової системи ext{2,3,4}
e2scrub_all	Перевіряє всі змонтовані файлові системи ext{2,3,4} на наявність помилок.
e2undo	Відтворює журнал скасування для файлової системи ext{2,3,4}, знайденої на пристрой. [Це можна використовувати для скасування невдалої операції, виконаної програмою E2fsprogs.]
e4crypt	Утиліта шифрування файлової системи ext4.
e4defrag	Онлайн-дефрагментатор для файлових систем ext4.
filefrag	Звіти про ступінь фрагментації певного файлу.
fsck.ext2	За замовчуванням перевіряє файлові системи ext2 і є жорстким посиланням на e2fsck .
fsck.ext3	За замовчуванням перевіряє файлові системи ext3 і є жорстким посиланням на e2fsck .
fsck.ext4	За замовчуванням перевіряє файлові системи ext4 і є жорстким посиланням на e2fsck .
logsave	Зберігає вихідні дані команди у файлі журналу.
lsattr	Виводить список атрибутів файлів у другій розширеній файловій системі.
mk_cmds	Перетворює таблицю імен команд і повідомлень довідки у файл вихідного коду C, придатний для використання з бібліотекою підсистеми lib
mke2fs	Створює файлову систему ext{2,3,4} на заданому пристрой
mkfs.ext2	За замовчуванням створює файлові системи ext2 і є жорстким посиланням на mke2fs
mkfs.ext3	За замовчуванням створює файлові системи ext3 і є жорстким посиланням на mke2fs
mkfs.ext4	За замовчуванням створює файлові системи ext4 і є жорстким посиланням на mke2fs
mklost+found	Створює каталог lost+found у файловій системі ext{2,3,4}; попередньо виділяє дискові блоки для цього каталогу, щоб полегшити завдання e2fsck
resize2fs	Може використовуватися для збільшення або зменшення файлових систем ext{2,3,4}
tune2fs	Налаштовує параметри файлової системи, що можна налаштовувати, у

файлових системах `ext{234}`.

`libcom_err` Загальна процедура відображення помилок.

`libe2p` Використовується **`dump2fs`, `chattr` та `lsattr`**.

`libext2fs` Містить процедури, що дозволяють програмам на рівні користувача маніпулювати файловими системами `ext{234}`.

`libss` Використовується **`debugfs`**.

8.81. Sysklogd-2.7.0

Пакет Sysklogd містить програми для реєстрації системних повідомлень, таких як ті, що виводяться ядром при незвичайних подіях.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 4.1 MB

8.81.1. Інсталяція Sysklogd

Підготуйте Sysklogd до компіляції:

```
./configure --prefix=/usr \
            --sysconfdir=/etc \
            --runstatedir=/run \
            --without-logger \
            --disable-static \
            --docdir=/usr/share/doc/sysklogd-2.7.0
```

Скомпілюйте пакет:

```
make
```

Цей пакет не містить набору тестів.

Встановіть пакет:

```
make install
```

8.81.2. Конфігурація Sysklogd

Створіть новий файл /etc/syslog.conf, виконавши наступне:

```
cat > /etc/syslog.conf << "EOF"
# Begin /etc/syslog.conf

auth,authpriv.* -/var/log/auth.log
*.authpriv.none -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *

# Do not open any internet ports.
secure_mode 2

# End /etc/syslog.conf
EOF
```

8.81.3. Вміст Sysklogd

Встановлені програми: syslogd

Короткі описи

syslogd

Реєструє повідомлення, які системні програми пропонують для реєстрації [Кожне зареєстроване повідомлення містить принаймні дату та ім'я хоста, а зазвичай також і назву програми, але це залежить від того, наскільки довіряє демон реєстрації.]

8.82. SysVinit-3.14

Пакет SysVinit містить програми для керування запуском, роботою та вимкненням системи.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 2.9 MB

8.82.1. Інсталяція SysVinit

Спочатку застосуйте патч, який видаляє кілька програм, встановлених іншими пакетами, уточнюює повідомлення та виправлює попередження компілятора:

```
patch -Np1 -i ../../sysvinit-3.14-consolidated-1.patch
```

Скомпілюйте пакет:

```
make
```

Цей пакет не містить набору тестів.

Встановіть пакет:

```
make install
```

8.82.2. Вміст SysVinit

Встановлені програми:	bootlogd, fstab-decode, halt, init, killall5, poweroff (посилання на halt), reboot (посилання на halt), runlevel, shutdown, and telinit (посилання на init)
------------------------------	---

Короткі описи

bootlogd	Записує повідомлення про завантаження в файл журналу
fstab-decode	Виконує команду з аргументами, закодованими за допомогою fstab
halt	Зазвичай викликає shutdown з опцією -h , але коли вже знаходиться в рівні запуску 0, він повідомляє ядро про необхідність зупинки системи; він записує в файл /var/log/wtmp , що система вимикається
init	Перший процес, що запускається після ініціалізації апаратного забезпечення ядром; він бере на себе процес завантаження і запускає всі процеси, вказані в його конфігураційному файлі
killall5	Відправляє сигнал всім процесам, крім процесів у власній сесії; він не вбиває батьківську оболонку
poweroff	Наказує ядру зупинити систему і вимкнути комп'ютер (див. halt)
reboot	Наказує ядру перезавантажити систему (див. halt)
runlevel	Повідомляє про попередній та поточний рівень запуску, як зазначено в останньому записі рівня запуску в /run/utmp
shutdown	Вимикає систему безпечним способом, сигналізуючи всім процесам та повідомляючи всіх користувачів, що ввійшли в систему
telinit	Повідомляє init , на який рівень запуску перейти

8.82. Про символи налагодження

Більшість програм і бібліотек за замовчуванням компілюються з включеними символами налагодження (за допомогою опції `-g gcc`). Це означає, що під час налагодження програми або бібліотеки, яка була скомпільована з інформацією про налагодження, налагоджувач може надавати не тільки адреси пам'яті, але й імена процедур і змінних.

Включення цих символів налагодження значно збільшує розмір програми або бібліотеки. Ось два приклади обсягу місця, яке займають ці символи:

- Бінарний файл **bash** із символами налагодження: 1200 КБ.
- Бінарний файл **bash** без символів налагодження: 480 КБ (на 60 % менше).
- Файли Glibc і GCC (`/lib` і `/usr/lib`) із символами налагодження: 87 МБ.
- Файли Glibc і GCC без символів налагодження: 16 МБ (на 82 % менше).

Розмір буде залежати від того, який компілятор і бібліотека С були використані, але програма, з якої були видалені символи налагодження, зазвичай на 50-80% менша за свою версію без видалення. Оскільки більшість користувачів ніколи не використовують налагоджувач у своєму системному програмному забезпеченні, видалення цих символів дозволяє звільнити багато місця на диску. У наступному розділі показано, як видалити всі символи налагодження з програм і бібліотек.

8.84. Видалення відлагоджувальної інформації

Цей розділ є необов'язковим. Якщо передбачуваний користувач не є програмістом і не планує виконувати будь-яке налагодження системного програмного забезпечення, розмір системи можна зменшити приблизно на 2 ГБ, видаливши символи налагодження та деякі непотрібні записи таблиці символів із бінарних файлів та бібліотек. Це не спричиняє жодних реальних незручностей для типового користувача Linux.

Більшість людей, які використовують команди, згадані нижче, не відчувають жодних труднощів. Однак легко зробити помилку і зробити нову систему непридатною для використання. Тому перед запуском команд **strip** рекомендується зробити резервну копію системи LFS у її поточному стані.

Команда **strip** з опцією `--strip-unneeded` видаляє всі символи налагодження з бінарного файлу або бібліотеки. Вона також видаляє всі записи таблиці символів, які не потрібні компілятору (для статичних бібліотек) або динамічному компілятору (для динамічно пов'язаних бінарних файлів і спільніх бібліотек).

Символи налагодження з вибраних бібліотек стискаються за допомогою Zlib і зберігаються в окремих файлах. Ця інформація про налагодження потрібна для подальшого виконання регресійних тестів за допомогою *valgrind* або *gdb* в BLFS.

Зверніть увагу, що **strip** перезапише бінарний файл або файл бібліотеки, який він обробляє. Це може привести до збою процесів, що використовують код або дані з цього файлу. Якщо процес, що виконує **strip**, буде порушений, бінарний файл або бібліотека, що обробляються, можуть бути знищені, що може зробити систему повністю непридатною для використання. Щоб уникнути цієї проблеми, ми копіюємо деякі бібліотеки та бінарні файли в `/tmp`, видаляємо їх там, а потім перевстановлюємо за допомогою команди **install**. (Відповідний запис у розділі 8.2.1, «Проблеми з оновленням», містить обґрунтування використання команди **install** у цьому випадку.)



Note

Ім'я завантажувача ELF — ld-linux-x86-64.so.2 у 64-роздрядних системах і ld-linux.so.2 у 32-роздрядних системах. Наведена нижче конструкція вибирає правильне ім'я для поточної архітектури, виключаючи все, що закінчується на `g`, у випадку, якщо наведені нижче команди вже були виконані.



Important

Якщо є якийсь пакет, версія якого відрізняється від версії, вказаної в книзі (або відповідно до рекомендацій з безпеки, або відповідно до особистих уподобань), може знадобитися оновити ім'я файлу бібліотеки в `save_usrlib` або `online_usrlib`. **Якщо цього не зробити, система може стати повністю непридатною для використання.**

```

save_usrlib=$(cd /usr/lib; ls ld-linux*[^g])
    libc.so.6
    libthread_db.so.1
    libquadmath.so.0.0.0
    libstdc++.so.6.0.33
    libitm.so.1.0.0
    libatomic.so.1.2.0"

cd /usr/lib

for LIB in $save_usrlib; do
    objcopy --only-keep-debug --compress-debug-sections=zlib $LIB $LIB.debug
    cp $LIB /tmp/$LIB
    strip --strip-unneeded /tmp/$LIB
    objcopy --add-gnu-debuglink=$LIB.debug /tmp/$LIB
    install -vm755 /tmp/$LIB /usr/lib
    rm /tmp/$LIB
done

online_usrbin="bash find strip"
online_usrlib="libbfd-2.44.so
                libsframe.so.1.0.0
                libhistory.so.8.2
                libncursesw.so.6.5
                libm.so.6
                libreadline.so.8.2
                libz.so.1.3.1
                libzstd.so.1.5.7
                $(cd /usr/lib; find libnss*.so* -type f)"

for BIN in $online_usrbin; do
    cp /usr/bin/$BIN /tmp/$BIN
    strip --strip-unneeded /tmp/$BIN
    install -vm755 /tmp/$BIN /usr/bin
    rm /tmp/$BIN
done

for LIB in $online_usrlib; do
    cp /usr/lib/$LIB /tmp/$LIB
    strip --strip-unneeded /tmp/$LIB
    install -vm755 /tmp/$LIB /usr/lib
    rm /tmp/$LIB
done

for i in $(find /usr/lib -type f -name '*.so*' ! -name '*dbg') \
        $(find /usr/lib -type f -name '*.a') \
        $(find /usr/{bin,sbin,libexec} -type f); do
    case "$online_usrbin $online_usrlib $save_usrlib" in
        *$(basename $i)* )
            ;;
        * ) strip --strip-unneeded $i
            ;;
    esac
done

unset BIN LIB save_usrlib online_usrbin online_usrlib

```

Велика кількість файлів буде позначена як помилки, оскільки їх формат не розпізнається. Ці попередження можна спокійно ігнорувати. Вони вказують на те, що ці файли є скриптами, а не бінарними файлами.

8.85. Очищення

Нарешті, очистіть деякі зайві файли, що залишилися після виконання тестів:

```
rm -rf /tmp/{*,.*}
```

У каталогах /usr/lib та /usr/libexec також є кілька файлів з розширенням .la. Це файли «libtool archive». У сучасній системі Linux файли libtool .la корисні лише для libltdl. Жодна бібліотека в LFS не повинна завантажуватися libltdl, і відомо, що деякі файли .la можуть порушити збірку пакетів BLFS. Видаліть ці файли зараз:

```
find /usr/lib /usr/libexec -name \*.la -delete
```

Для отримання додаткової інформації про файли архіву libtool див. розділ BLFS «Про файли архіву Libtool (.la)».

Компілятор, створений у розділах 6 та 7, все ще частково встановлений і більше не потрібен. Видаліть його за допомогою:

```
find /usr -depth -name $(uname -m)-lfs-linux-gnu\* | xargs rm -rf
```

Нарешті, видаліть тимчасовий обліковий запис користувача «tester», створений на початку попереднього розділу.

```
userdel -r tester
```

Глава 9. Конфігурація системи

9.1. Вступ

Завантаження системи Linux включає в себе кілька завдань. Процес повинен змонтувати як віртуальні, так і реальні файлові системи, ініціалізувати пристрої, перевірити цілісність файлових систем, змонтувати та активувати будь-які розділи або файли підкачки, встановити системний годинник, запустити мережу, запустити будь-які демони, необхідні системі, та виконати будь-які інші завдання, визначені користувачем. Цей процес повинен бути організований таким чином, щоб завдання виконувалися в правильному порядку та якомога швидше.

9.1.1. System V

System V — це класичний процес завантаження, який використовується в Unix та Unix-подібних системах, таких як Linux, приблизно з 1983 року. Він складається з невеликої програми **init**, яка налаштовує основні процеси, такі як **login** (через getty), та запускає скрипт. Цей скрипт, зазвичай названий **rc**, контролює виконання набору додаткових скриптів, які виконують завдання, необхідні для ініціалізації системи.

Програма **init** керується файлом `/etc/inittab` і організована в рівні запуску, які може вибирати користувач. У LFS вони використовуються наступним чином:

```
0 - halt
1 - Single user mode
2 - User definable
3 - Full multiuser mode
4 - User definable
5 - Full multiuser mode with display manager
6 - reboot
```

Звичайний рівень запуску за замовчуванням — 3 або 5.

Переваги

- Встановлена, добре зрозуміла система.
- Легко налаштовується.

Недоліки

- Може завантажуватися повільніше. Середньошвидкісна базова система LFS завантажується за 8-12 секунд, де час завантаження вимірюється від першого повідомлення ядра до запиту на вхід. Зв'язок з мережею зазвичай встановлюється приблизно через 2 секунди після запиту на вхід.
- Послідовна обробка завдань завантаження. Це пов'язано з попереднім пунктом. Затримка в будь-якому процесі, такому як перевірка файлової системи, затримає весь процес завантаження.
- Не підтримує безпосередньо такі розширені функції, як групи контролю (cgroups) та справедливе розподілення ресурсів між користувачами.
- Додавання скриптів вимагає ручного, статичного визначення послідовності.

9.2. LFS-Bootscripts-20240825

Пакет LFS-Bootscripts містить набір скриптів для запуску/зупинки системи LFS під час завантаження/виключення. Конфігураційні файли та процедури, необхідні для налаштування процесу завантаження, описані в наступних розділах.

Приблизний час побудови: менше ніж 0.1 SBU

Необхідний простір на диску: 244 KB

9.2.1. Інсталляція LFS-Bootscripts

Встановіть пакет:

```
make install
```

9.2.2. Вміст LFS-Bootscripts

Встановлені скрипти:	checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountvirtfs, network, rc, reboot, sendsignals, setclock, ipv4-static, swap, sysctl, sysklogd, template, udev, and udev_retry
Встановлені директорії:	/etc/rc.d, /etc/init.d (символічне посилання), /etc/sysconfig, /lib/services, /lib/lsb (символічне посилання)

Короткі описи

checkfs	Перевіряє цілісність файлових систем перед їх монтуванням (за винятком журнальних та мережевих файлових систем)
cleanfs	Видаляє файли, які не повинні зберігатися між перезавантаженнями, такі як файли в /run/ та /var/lock/; відновлює /run/utmp та видаляє файли /etc/nologin, /fastboot та /forcefsck, які можуть бути присутніми
console	Завантажує правильну таблицю розкладки клавіатури для бажаної розкладки клавіатури; також встановлює шрифт екрану
functions	Містить загальні функції, такі як перевірка помилок і стану, які використовуються декількома скриптами завантаження
halt	Зупиняє систему
ifdown	Зупиняє мережевий пристрій
ifup	Ініціалізує мережевий пристрій
localnet	Встановлює ім'я хоста системи та локальний пристрій зворотного зв'язку
modules	Завантажує модулі ядра, перелічені в /etc/sysconfig/modules, використовуючи аргументи, які також вказані там
mountfs	Монтує всі файлові системи, крім тих, що позначені як noauto або є мережевими
mountvirtfs	Монтує віртуальні файлові системи ядра, такі як proc

network	Налаштовує мережеві інтерфейси, такі як мережеві карти, та встановлює шлюз за замовчуванням (де це можливо)
rc	Головний скрипт управління рівнем запуску; він відповідає за послідовне виконання всіх інших скриптів завантаження, у порядку, визначеному іменами символічних посилань на ці інші скрипти завантаження
reboot	Перезавантажує систему
sendsignals	Перевіряє, чи всі процеси завершені перед перезавантаженням або зупинкою системи
setclock	Скидає системний годинник до місцевого часу, якщо апаратний годинник не налаштований на UTC
ipv4-static	Надає функціональність, необхідну для призначення статичної адреси Інтернет-протоколу (IP) мережевому інтерфейсу
swap	Увімкнення та вимкнення файлів і розділів підкачки.
sysctl	Завантаження значень конфігурації системи з файлу <code>/etc/sysctl.conf</code> , якщо такий файл існує, у запущене ядро.
sysklogd	Запуск і зупинка системних і ядерових демонів журналів.
template	Шаблон для створення власних скриптів завантаження для інших демонів.
udev	Підготовка каталогу <code>/dev</code> і запуск демона <code>udev</code> .
udev_retry	Повторює невдалі події <code>udev</code> і копіює згенеровані файли правил з <code>/run/udev</code> до <code>/etc/udev/rules.d</code> , якщо потрібно

9.3. Огляд роботи з пристроями та модулями

У розділі 8 ми встановили демон `udev` під час компіляції `udev`. Перш ніж перейти до детального опису роботи `udev`, варто коротко розглянути історію попередніх методів роботи з пристроями.

Системи Linux загалом традиційно використовували статичний метод створення пристрій, за допомогою якого під `/dev` створювалося велика кількість вузлів пристрій (іноді буквально тисячі вузлів), незалежно від того, чи існували відповідні апаратні пристрої насправді. Зазвичай це робилося за допомогою скрипта **MAKEDEV**, який містив низку викликів програми `mknod` з відповідними основними та додатковими номерами пристрій для кожного можливого пристрою, який міг існувати у світі.

За допомогою методу `udev` вузли пристрій створюються тільки для тих пристрій, які виявлені ядром. Ці вузли пристрій створюються кожного разу під час завантаження системи; вони зберігаються у файловій системі `devtmpfs` (віртуальний файловий системі, яка повністю знаходиться в системній пам'яті). Вузли пристрій не вимагають багато місця, тому використовувана пам'ять є незначною.

9.3.1. Історія

У лютому 2000 року нова файлова система під назвою `devfs` була інтегрована в ядро 2.3.46 і стала доступною в серії стабільних ядер 2.4. Хоча вона була присутня в самому вихідному коді ядра, цей метод динамічного створення пристрій ніколи не отримав переважної підтримки з боку основних розробників ядра.

Головною проблемою підходу, прийнятого `devfs`, був спосіб обробки виявлення, створення та іменування пристрій. Остання проблема, пов'язана з іменуванням вузлів пристрій, була, мабуть, найкритичнішою. Загальноприйнято, що якщо імена пристрій є конфігурованими, політика іменування пристрій повинна обиратися системними адміністраторами, а не нав'язуватися їм розробниками. Файлова система `devfs` також страждала від умов гонки, що були властиві її дизайну; їх не можна було віправити без істотної переробки ядра. `devfs` довгий час вважався застарілим і, зрештою, був вилучений з ядра в червні 2006 року.

З розвитком нестабільного дерева ядра 2.5, яке пізніше було випущено як серія стабільних ядер 2.6, з'явилася нова віртуальна файлова система під назвою `sysfs`. Завданням `sysfs` є надання інформації про апаратну конфігурацію системи процесам у просторі користувача. Завдяки цій видимій у просторі користувача репрезентації стало можливим розробити заміну `devfs` у просторі користувача.

9.3.2. Впровадження Udev

9.3.2.1. Sysfs

Вище було коротко згадано про файлову систему `sysfs`. Можна задатися питанням, як `sysfs` дізнається про пристрій, присутні в системі, і які номери пристрій слід використовувати для них. Драйвери, які були скомпільовані в ядро, реєструють свої об'єкти в `sysfs` (внутрішньо `devtmpfs`) у міру їх виявлення ядром. Для драйверів, скомпільованих як модулі, реєстрація відбувається під час завантаження модуля. Після монтування файлової системи `sysfs` (у `/sys`) дані, які драйвери зареєстрували в `sysfs`, стають доступними для процесів у просторі користувача та для `udevd` для обробки (включно з модифікаціями вузлів пристрій).

9.3.2.2. Створення вузла пристрою

Файли пристроїв створюються ядром у файловій системі `devtmpfs`. Будь-який драйвер, який бажає зареєструвати вузол пристрою, використовує `devtmpfs` (через ядро драйвера) для цього. Коли екземпляр `devtmpfs` монтується на `/dev`, вузол пристрою спочатку буде відкритий для користувачького простору з фіксованим іменем, правами доступу та власником.

Через короткий час ядро надішло uevent до `udevd`. На основі правил, зазначених у файлах у каталогах `/etc/udev/rules.d`, `/usr/lib/udev/rules.d` та `/run/udev/rules.d`, `udevd` створить додаткові символічні посилання на вузол пристрою, або змінить його дозволи, власника чи групу, або модифікує внутрішній запис у базі даних `udevd` (ім'я) для цього об'єкта.

Правила в цих трьох каталогах пронумеровані, і всі три каталоги об'єднані разом. Якщо `udevd` не може знайти правило для пристрою, який він створює, він залишить дозволи та права власності такими, якими вони були спочатку в `devtmpfs`.

9.3.2.3. Завантаження модуля

Драйвери пристроїв, скомпільовані як модулі, можуть мати вбудовані псевдоніми. Псевдоніми відображаються у вихідних даних програми `modinfo` і зазвичай пов'язані з ідентифікаторами пристроїв, що підтримуються модулем, для конкретної шини. Наприклад, драйвер `snd-fm801` підтримує пристрій PCI з ідентифікатором постачальника `0x1319` та ідентифікатором пристрою `0x0801` і має псевдонім `pci:v00001319d00000801sv*sd*bc04sc01i*`. Для більшості пристроїв драйвер шини експортує псевдонім драйвера, який обробляє пристрій через `sysfs`. Наприклад, файл `/sys/bus/pci/devices/0000:00:0d.0/modalias` може містити рядок `pci:v00001319d00000801sv00001319sd00001319bc04sc01i00`. Правила за замовчуванням, що надаються з `udev`, змусять `udevd` викликати `/sbin/modprobe` із вмістом змінної середовища `MODALIAS` `uevent` (яка повинна бути такою ж, як і вміст файла `modalias` у `sysfs`), таким чином завантажуючи всі модулі, псевдоніми яких відповідають цьому рядку після розширення символів-замінників

У цьому прикладі це означає, що, крім `snd-fm801`, буде завантажено застарілий (і непотрібний) драйвер `forte`, якщо він доступний. Дивіться нижче, як можна запобігти завантаженню непотрібних драйверів.

Ядро також може завантажувати модулі для мережевих протоколів, файлових систем та підтримки NLS за запитом.

9.3.2.4. Робота з гаряче-підключеними/динамічними пристроями

Коли ви підключаете пристрій, наприклад MP3-плеєр з інтерфейсом Universal Serial Bus (USB), ядро розпізнає, що пристрій підключено, і генерує uevent. Цей uevent потім обробляється `udevd`, як описано вище.

9.3.3. Проблеми із завантаженням модулів та створенням пристройв

Існує кілька можливих проблем, пов'язаних з автоматичним створенням вузлів пристройв.

9.3.3.1. Модуль ядра не завантажується автоматично

Udev завантажить модуль тільки в тому випадку, якщо він має специфічний для шини псевдонім, а драйвер шини правильно експортує необхідні псевдоніми в sysfs. В інших випадках завантаження модуля слід організувати іншими засобами. Відомо, що в Linux-6.13.4 udev завантажує правильно написані драйвери для пристрій INPUT, IDE, PCI, USB, SCSI, SERIO та FireWire.

Щоб визначити, чи має необхідний драйвер пристрою необхідну підтримку для udev, запустіть **modinfo** з іменем модуля як аргументом. Тепер спробуйте знайти каталог пристрою в `/sys/bus` і перевірте, чи є там файл `modalias`.

Якщо файл `modalias` існує в `sysfs`, драйвер підтримує пристрій і може безпосередньо з ним спілкуватися, але не має псевдоніма, це є помилкою в драйвері. Завантажте драйвер без допомоги udev і очікуйте, що проблема буде виправлена пізніше.

Якщо у відповідному каталозі під `/sys/bus` немає файлу `modalias`, це означає, що розробники ядра ще не додали підтримку `modalias` для цього типу шини. У Linux-6.13.4 це стосується шин ISA. Очікуйте, що ця проблема буде вирішена в наступних версіях ядра.

Udev не призначений для завантаження «обгорткових» драйверів, таких як `snd-pcm-oss`, та неапаратних драйверів, таких як `loop`.

9.3.3.2. Модуль ядра не завантажується автоматично, і Udev не призначений для його завантаження

Якщо модуль «обгортка» лише розширює функціональність, що надається іншим модулем (наприклад, `snd-pcm-oss` розширює функціональність `snd-pcm`, роблячи звукові карти доступними для програм OSS), налаштуйте **modprobe** так, щоб він завантажував обгортку після того, як udev завантажить модуль, що обгортяється. Для цього додайте рядок `«softdep»` до відповідного файлу `/etc/modprobe.d/<filename>.conf`. Наприклад:

```
softdep snd-pcm post: snd-pcm-oss
```

Зверніть увагу, що команда `«softdep»` також дозволяє використовувати залежності `pre:` або поєднання залежностей `pre: i post::`. Докладнішу інформацію про синтаксис і можливості `«softdep»` див. у довідковій сторінці *modprobe.d(5)*.

Якщо модуль, про який йде мова, не є обгорткою і є корисним сам по собі, налаштуйте скрипт завантаження **modules**, щоб завантажувати цей модуль під час завантаження системи. Для цього додайте ім'я модуля до файлу `/etc/sysconfig/modules` в окремому рядку. Це працює і для модулів-обгортків, але в цьому випадку є неоптимальним.

9.3.3.3. Udev завантажує деякі небажані модулі

Або не створюйте модуль, або внесіть його до чорного списку у файлі `/etc/modprobe.d/blacklist.conf`, як це зроблено з модулем `forte` у прикладі нижче:

```
blacklist forte
```

Модулі, що знаходяться в чорному списку, все ще можна завантажити вручну за допомогою команди **modprobe**.

9.3.3.4. Udev створює пристрій неправильно або створює неправильне символічне посилання

Зазвичай це трапляється, якщо правило несподівано збігається з пристроєм. Наприклад, неправильно написане правило може збігатися як із диском SCSI (як потрібно), так і з відповідним загальним пристроєм SCSI (неправильно) за виробником. Знайдіть правило, що викликає проблему, і зробіть його більш конкретним за допомогою команди `udevadm info`.

9.3.3.5. Правило Udev працює ненадійно

Це може бути ще одним проявом попередньої проблеми. Якщо ні, і ваше правило використовує атрибути `sysfs`, це може бути проблема з синхронізацією ядра, яка буде виправлена в наступних версіях ядра. Наразі ви можете обійти цю проблему, створивши правило, яке чекає на використаний атрибут `sysfs`, і додавши його до файлу `/etc/udev/rules.d/10-wait_for_sysfs.rules` (створіть цей файл, якщо він не існує). Будь ласка, повідомте про це список розробників LFS, якщо ви це зробите і це допоможе.

9.3.3.6. Udev не створює пристрій

Спочатку переконайтесь, що драйвер вбудований в ядро або вже завантажений як модуль, і що udev не створює неправильно названий пристрій.

Якщо драйвер ядра не експортує свої дані в `sysfs`, udev не має інформації, необхідної для створення вузла пристрою. Це найчастіше трапляється з драйверами сторонніх виробників, які не входять до дерева ядра. Створіть статичний вузол пристрою в `/usr/lib/udev/devices` з відповідними `major/minor` номерами (див. файл `devices.txt` у документації ядра або документацію, надану стороннім постачальником драйвера). Статичний вузол пристрою буде скопійований в `/dev` за допомогою `udev`.

9.3.3.7. Порядок іменування пристріїв змінюється випадково після перезавантаження

Це пов'язано з тим, що udev, за своїм дизайном, обробляє `uevents` і завантажує модулі паралельно, і, отже, в непередбачуваному порядку. Це ніколи не буде «відремонтовано». Не слід покладатися на стабільність імен пристріїв ядра. Натомість створіть власні правила, які створюють символічні посилання зі стабільними іменами на основі деяких стабільних атрибутів пристрою, таких як серійний номер або вихідні дані різних утиліт `*_id`, встановлених udev. Приклади див. у розділі 9.4, «Управління пристроями», та розділі 9.5, «Загальна конфігурація мережі».

9.3.4. Корисно до прочитання

Додаткова корисна документація доступна на наступних сайтах:

- Реалізація `devfs` у просторі користувача http://www.kroah.com/linux/talks/ols_2003_udc_paper/Reprint-Kroah-Hartman-OLS2003.pdf
- Файлова система `sysfs` <https://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf>

9.4. Управління пристроями

9.4.1. Мережеві пристрої

Udev, за замовчуванням, називає мережеві пристрої відповідно до даних прошивки/BIOS або фізичних характеристик, таких як шина, слот або MAC-адреса. Мета цієї конвенції іменування — забезпечити послідовність іменування мережевих пристроїв, а не базуватися на тому, коли була виявлена мережева карта. У старіших версіях Linux — наприклад, на комп'ютері з двома мережевими картами, виготовленими Intel і Realtek — мережева карта, виготовлена Intel, могла стати eth0, а карта Realtek — eth1. Після перезавантаження карти іноді перенумеровувалися навпаки.

У новій схемі іменування типові імена мережевих пристроїв мають вигляд `enp5s0` або `wlp3s0`. Якщо така схема іменування не підходить, можна застосувати традиційну схему іменування або власну схему.

9.4.1.1. Вимкнення постійного іменування в командному рядку ядра

Традиційну схему іменування з використанням `eth0`, `eth1` тощо можна відновити, додавши `net.ifnames=0` у командний рядок ядра. Це найбільш підходить для систем, які мають лише один пристрій Ethernet певного типу. Ноутбуки часто мають два підключення Ethernet з іменами `eth0` та `wlan0`; такі ноутбуки також можуть використовувати цей метод. Командний рядок знаходитьться у файлі конфігурації GRUB. Див. Розділ 10.4.4, «Створення файлу конфігурації GRUB».

9.4.1.2. Створення власних правил Udev

Схему іменування можна налаштувати, створивши власні правила udev. Додано скрипт, який генерує початкові правила. Згенеруйте ці правила, виконавши:

```
bash /usr/lib/udev/init-net-rules.sh
```

Тепер перевірте файл `/etc/udev/rules.d/70-persistent-net.rules`, щоб дізнатися, яке ім'я було присвоєно якому мережевому пристрою:

```
cat /etc/udev/rules.d/70-persistent-net.rules
```



Note

У деяких випадках, наприклад, коли MAC-адреси були призначені мережевій карті вручну або у віртуальному середовищі, такому як Qemu або Xen, файл мережевих правил може не бути створений, оскільки адреси не призначаються послідовно. У таких випадках цей метод не може бути використаний.

Файл починається з блоку коментарів, за яким йдуть два рядки дляожної мережевої карти. Перший рядок дляожної мережевої карти є коментарем, що містить її апаратні ідентифікатори (наприклад, ідентифікатори виробника та пристрою PCI, якщо це PCI-карта), а також її драйвер (у дужках, якщо драйвер можна знайти). Апаратний ідентифікатор та драйвер не використовуються для визначення імені, яке слід присвоїти інтерфейсу; ця інформація є лише довідковою. Другий рядок — це правило udev, яке відповідає цій мережевій карті і фактично присвоює їй ім'я.

Всі правила udev складаються з декількох ключових слів, розділених комами та необов'язковими пробілами. Ось ці ключові слова та пояснення кожного з них:

- SUBSYSTEM==«net» - Це вказує udev ігнорувати пристрой, які не є мережевими картами.
- ACTION==«add» - Це вказує udev ігнорувати це правило для uevent, який не є додаванням ("remove" і «change» uevents також трапляються, але не потребують перейменування мережевих інтерфейсів).
- DRIVERS==«?*» - Це існує для того, щоб udev ігнорував підінтерфейси VLAN або bridge (оскільки ці підінтерфейси не мають драйверів). Ці підінтерфейси пропускаються, оскільки ім'я, яке було б присвоєно, конфліктувало б з батьківськими пристроями.
- ATTR{address} - Значенням цього ключового слова є MAC-адреса мережової карти.
- ATTR{type}==«1» - Це гарантує, що правило буде відповідати тільки основному інтерфейсу в разі певних бездротових драйверів, які створюють кілька віртуальних інтерфейсів. Додаткові інтерфейси пропускаються з тієї ж причини, що і підінтерфейси VLAN та bridge: в іншому випадку відбудеться конфлікт імен.
- NAME - Значенням цього ключового слова є ім'я, яке udev присвоїть цьому інтерфейсу.

Значення NAME є важливою частиною. Перед тим, як продовжувати, переконайтесь, що ви знаєте, яке ім'я було присвоєно кожній з ваших мережевих карт, і обов'язково використовуйте це значення NAME при створенні файлів конфігурації мережі.

Навіть якщо файл правил udev створено, udev все одно може присвоїти одну або кілька альтернативних назв для мережової карти на основі фізичних характеристик. Якщо власне правило udev перейменує деякі мережеві карти, використовуючи ім'я, яке вже призначено як альтернативне ім'я іншої мережової карти, це правило udev не спрацює. Якщо така проблема виникне, ви можете створити файл конфігурації посилання /etc/udev/network/99-default. з порожньою політикою призначення альтернативних імен, замінивши файл конфігурації за замовчуванням /usr/lib/udev/network/99-default.link:

```
sed -e '/^AlternativeNamesPolicy/s/=.*$/=/' \
      /usr/lib/udev/network/99-default.link \
> /etc/udev/network/99-default.link
```

9.4.2. CD-ROM символічні посилання

Деякі програми, які ви, можливо, захотите встановити пізніше (наприклад, різні медіаплеєри), очікують, що символільні посилання /dev/cdrom та /dev/dvd існують і вказують на пристрій CD-ROM або DVD-ROM. Також може бути зручно помістити посилання на ці символільні посилання в /etc/fstab. Udev постачається зі скриптом, який генерує файли правил для створення цих символічних посилань залежно від можливостей кожного пристрою, але вам потрібно вирішити, який із двох режимів роботи ви хочете використовувати в скрипті.

По-перше, скрипт може працювати в режимі «by-path» (використовується за замовчуванням для пристройв USB і FireWire), де правила, які він створює, залежать від фізичного шляху до пристроя CD або DVD. Подруге, він може працювати в режимі «by-id» (за замовчуванням для пристройв IDE і SCSI), де правила, які він створює, залежать від ідентифікаційних рядків, що зберігаються на самому пристрой CD або DVD. Шлях

визначається скриптом **path_id** udev, а ідентифікаційні рядки зчитуються з апаратного забезпечення програмами **ata_id** або **scsi_id**, залежно від типу вашого пристрію.

Кожен підхід має свої переваги; правильний підхід залежить від того, які зміни можуть відбутися в пристрії. Якщо ви очікуєте, що фізичний шлях до пристрію (тобто порти та/або слоти, до яких він підключається) зміниться, наприклад, тому що ви плануєте перемістити диск до іншого порту IDE або іншого роз'єму USB, то вам слід використовувати режим «*by-id*». З іншого боку, якщо ви очікуєте, що ідентифікатор пристрію зміниться, наприклад, через його вихід з ладу, і ви плануєте замінити його іншим пристроям, який підключається до тих самих роз'ємів, то вам слід використовувати режим «*by-path*».

Якщо з вашим приводом можливий будь-який з цих типів змін, виберіть режим, виходячи з того, який тип змін, на вашу думку, буде відбуватися частіше.



Important

Зовнішні пристрії (наприклад, CD-привід, підключений через USB) не повинні використовувати стійкість за шляхом, оскільки кожного разу, коли пристрій підключається до нового зовнішнього порту, його фізичний шлях змінюється. Усі зовнішні пристрії матимуть цю проблему, якщо ви напишете правила udev для їх розпізнавання за фізичним шляхом; проблема не обмежується CD-та DVD-приводами.

Якщо ви хочете побачити значення, які будуть використовувати скрипти udev, то для відповідного пристроя CD-ROM знайдіть відповідний каталог у `/sys` (наприклад, це може бути `/sys/block/hdd`) і виконайте команду, подібну до наступної:

```
udevadm test /sys/block/hdd
```

Подивітесь на рядки, що містять вихідні дані різних програм `*_id`. У режимі «*by-id*» буде використовуватися значення `ID_SERIAL`, якщо воно існує і не є порожнім, інакше буде використовуватися комбінація `ID_MODEL` та `ID_REVISION`. У режимі «*by-path*» буде використовуватися значення `ID_PATH`.

Якщо режим за замовчуванням не підходить для вашої ситуації, можна внести наступні зміни до файлу `/etc/udev/rules.d/83-cdrom-symlinks.rules`, як показано нижче (де `mode` — це «*by-id*» або «*by-path*»):

```
sed -e 's/"write_cd_rules"/"write_cd_rules mode"/' \
-i /etc/udev/rules.d/83-cdrom-symlinks.rules
```

Зверніть увагу, що на даний момент не потрібно створювати файли правил або символльні посилання, оскільки ви підключили каталог `/dev` хоста до системи LFS і ми припускаємо, що символльні посилання існують на хості. Правила та символльні посилання будуть створені під час першого запуску системи LFS.

Однак, якщо у вас є кілька CD-ROM-пристроїв, то символльні посилання, створені в цей час, можуть вказувати на інші пристрії, ніж ті, на які вони вказують на вашому хості, оскільки пристрії не виявляються в передбачуваному порядку. Призначення, створені під час першого завантаження системи LFS, будуть стабільними, тому це є проблемою лише в тому випадку, якщо вам потрібно, щоб символльні посилання на обох системах вказували на один і той самий пристрій. Якщо вам це потрібно, то після завантаження перевірте (і, можливо, відредактуйте) створений файл `/etc/udev/rules.d/70-persistent-cd.rules`, щоб переконатися, що призначені символльні посилання відповідають вашим потребам.

9.4.3. Робота з дублікатами пристройв

Як пояснено в розділі 9.3, «Огляд роботи з пристроями та модулями», порядок, в якому пристрой з однаковою функцією з'являється в `/dev`, є в основному випадковим. Наприклад, якщо у вас є USB-вебкамера та ТВ-тюнер, іноді `/dev/video0` посилається на камеру, а `/dev/video1` — на тюнер, а іноді після перезавантаження порядок змінюється. Для всіх класів апаратного забезпечення, крім звукових карт і мережевих карт, це можна віправити, створивши правила `udev` для створення постійних символічних посилань. Випадок мережевих карт розглядається окремо в розділі 9.5, «Загальна конфігурація мережі», а конфігурацію звукової карти можна знайти в BLFS.

Для кожного з ваших пристройв, на якому може виникнути ця проблема (навіть якщо вона не існує у вашій поточній дистрибуції Linux), знайдіть відповідний каталог у `/sys/class` або `/sys/block`. Для відеопристроїв це може бути `/sys/class/video4linux/videoX`. Визначте атрибути, які однозначно ідентифікують пристрой (зазвичай це ідентифікатори виробника та продукту та/або серійні номери):

```
udevadm info -a -p /sys/class/video4linux/video0
```

Потім напишіть правила, які створюють символічні посилання, наприклад:

```
cat > /etc/udev/rules.d/83-duplicate_devs.rules << "EOF"

# Persistent symlinks for webcam and tuner
KERNEL=="video*", ATTRS{idProduct}=="1910", ATTRS{idVendor}=="0d81", SYMLINK+="webcam"
KERNEL=="video*", ATTRS{device}=="0x036f", ATTRS{vendor}=="0x109e", SYMLINK+="tvtuner"

EOF
```

В результаті пристрой `/dev/video0` та `/dev/video1` все ще випадково посилаються на тюнер та веб-камеру (і тому ніколи не повинні використовуватися безпосередньо), але існують символічні посилання `/dev/tvtuner` та `/dev/webcam`, які завжди вказують на правильний пристрой.

9.5. Загальна конфігурація мережі

9.5.1. Створення файлів конфігурації мережевого інтерфейсу

Файли в `/etc/sysconfig/` зазвичай визначають, які інтерфейси підключаються та відключаються мережевим скриптом. Цей каталог повинен містити файл для кожного інтерфейсу, що налаштовується, наприклад `ifconfig.xuz`, де «`xuz`» описує мережеву карту. Зазвичай підходить ім'я інтерфейсу (наприклад, `eth0`). Кожен файл містить атрибути одного інтерфейсу, такі як його IP-адреса (адреси), маски підмережі тощо. Корінь імені файлу повинен бути `ifconfig`.

Note

Якщо процедура, описана в попередньому розділі, не була використана, udev призначить імена інтерфейсів мережевих карт на основі фізичних характеристик системи, таких як enp2s1. Якщо ви не впевнені, яке ім'я має ваш інтерфейс, ви завжди можете запустити **ip link** або **ls /sys/class/net** після завантаження системи.

Імена інтерфейсів залежать від реалізації та конфігурації демона udev, що працює в системі. Демон udev для LFS (встановлений у розділі 8.76, «Udev з Systemd-257.3») не буде працювати, поки система LFS не завантажиться. Тому імена інтерфейсів у системі LFS не завжди можна визначити, запустивши ці команди на хост-дистрибутиві, *навіть у середовищі chroot*.

Наступна команда створює зразок файлу для пристрою eth0 зі статичною IP-адресою:

```
cd /etc/sysconfig/
cat > ifconfig.eth0 << "EOF"
ONBOOT=yes
IFACE=eth0
SERVICE=ipv4-static
IP=192.168.1.2
GATEWAY=192.168.1.1
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

Значення, виділені курсивом, необхідно змінити в кожному файлі, щоб правильно налаштувати інтерфейси.

Якщо змінна `ONBOOT` встановлена на `yes`, мережевий скрипт System V запустить мережеву карту (NIC) під час процесу завантаження системи. Якщо встановлено будь-яке інше значення, крім `yes`, мережевий скрипт проігнорує мережеву карту і вона не буде запущена автоматично. Інтерфейси можна запустити або зупинити вручну за допомогою команд **ifup** та **ifdown**.

Змінна `IFACE` визначає ім'я інтерфейсу, наприклад, `eth0`. Вона необхідна для всіх файлів конфігурації мережевих пристрій. Розширення імені файлу повинно відповідати цьому значенню.

Змінна `SERVICE` визначає метод, що використовується для отримання IP-адреси. Пакет LFS-Bootscripts має модульний формат призначення IP-адрес, а створення додаткових файлів у каталозі `/lib/services/` дозволяє використовувати інші методи призначення IP-адрес. Це зазвичай використовується для протоколу динамічної конфігурації хоста (DHCP), який розглядається в книзі BLFS.

Змінна `GATEWAY` повинна містити IP-адресу шлюзу за замовчуванням, якщо така є. Якщо її немає, то повністю закомментуйте цю змінну.

Змінна `PREFIX` визначає кількість бітів, що використовуються в підмережі. Кожен сегмент IP-адреси має 8 бітів. Якщо маска підмережі дорівнює 255.255.255.0, то для визначення номера мережі використовуються перші три сегменти (24 біти). Якщо маска підмережі дорівнює 255.255.255.240, підмережа використовує перші 28 бітів. Префікси довжиною більше 24 бітів зазвичай використовуються DSL та кабельними провайдерами інтернет-послуг (ISP). У цьому прикладі (`PREFIX=24`) маска мережі дорівнює 255.255.255.0.

Налаштуйте змінну PREFIX відповідно до вашої підмережі. Якщо її не вказано, PREFIX за замовчуванням дорівнює 24.

Для отримання додаткової інформації див. сторінку `man ifup`.

9.5.2. Створення файлу /etc/resolv.conf

Система потребуватиме певних засобів для отримання розпізнавання імен доменної служби (DNS) для перетворення імен інтернет-доменів в IP-адреси і навпаки. Найкраще це зробити, вказавши IP-адресу DNS-сервера, яку можна отримати у інтернет-провайдера або адміністратора мережі, у файлі `/etc/resolv.conf`. Створіть файл, виконавши наступну команду:

```
cat > /etc/resolv.conf << "EOF"
# Begin /etc/resolv.conf

domain <Your Domain Name>
nameserver <IP address of your primary nameserver>
nameserver <IP address of your secondary nameserver>

# End /etc/resolv.conf
EOF
```

Заяву про `domain` можна опустити або замінити заявою `search`. Дивіться сторінку довідки для `resolv.conf` для більш детальної інформації.

Замініть `<IP address of your primary nameserver>` на IP-адресу DNS, яка найбільш підходить для налаштування. Часто буває більше одного запису (вимоги вимагають наявності вторинних серверів для резервного копіювання). Якщо вам потрібен або ви хочете мати тільки один DNS-сервер, видаліть другий рядок `nameserver` з файлу. IP-адреса також може бути маршрутизатором у локальній мережі.



Публічні IPv4-адреси DNS Google — 8.8.8.8 та 8.8.4.4.

9.5.3. Налаштування імені хоста системи

Під час процесу завантаження файл `/etc/hostname` використовується для встановлення імені хоста системи.

Створіть файл `/etc/hostname` і введіть ім'я хоста, виконавши:

```
echo "<lfs>" > /etc/hostname
```

`<lfs>` потрібно замінити на ім'я, присвоєне комп'ютеру. Не вводьте тут повне доменне ім'я (FQDN). Ця інформація вноситься у файл `/etc/hosts`.

9.5.4. Налаштування файлу /etc/hosts

Виберіть повне доменне ім'я (FQDN) та можливі псевдоніми для використання у файлі `/etc/hosts`. Якщо ви використовуєте статичні IP-адреси, вам також потрібно буде вибрати IP-адресу. Синтаксис запису у файлі `hosts` такий:

```
IP_address myhost.example.org aliases
```

Якщо комп'ютер не буде доступний в Інтернеті (тобто, якщо немає зареєстрованого домену та дійсного блоку призначених IP-адрес — більшість користувачів цього не мають), переконайтесь, що IP-адреса знаходиться в діапазоні IP-адрес приватної мережі. Дійсні діапазони:

Private Network Address Range	Normal Prefix
10.0.0.1 – 10.255.255.254	8
172.x.0.1 – 172.x.255.254	16
192.168.y.1 – 192.168.y.254	24

x може бути будь-яким числом в діапазоні від 16 до 31. y може бути будь-яким числом в діапазоні від 0 до 255.

Дійсною приватною IP-адресою може бути 192.168.1.2.

Якщо комп'ютер повинен бути видимим в Інтернеті, дійсним FQDN може бути саме доменне ім'я або рядок, отриманий шляхом об'єднання префікса (часто імені хоста) та доменного імені за допомогою символу «.». Крім того, вам потрібно зв'язатися з провайдером домену, щоб вирішити FQDN на вашу публічну IP-адресу.

Навіть якщо комп'ютер не видно в Інтернеті, FQDN все одно необхідний для нормальної роботи деяких програм, таких як MTA. Для цієї мети можна використовувати спеціальний FQDN, localhost.localdomain.

Створіть файл /etc/hosts, виконавши:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.1.2> <FQDN> <HOSTNAME> [alias1] [alias2 ...]
::1      localhost ip6-localhost ip6-loopback
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

# End /etc/hosts
EOF
```

Значення <192.168.1.2>, <FQDN> та <HOSTNAME> потрібно змінити для конкретних випадків використання або вимог (якщо IP-адреса призначена мережевим/системним адміністратором і комп'ютер буде підключений до існуючої мережі). Опціональні псевдоніми можна пропустити.

9.6. Використання та конфігурація скрипту завантаження System V

9.6.1. Як працюють скрипти завантаження System V?

Ця версія LFS використовує спеціальний завантажувальний механізм під назвою SysVinit, заснований на серії *levels* запуску. Процедура завантаження може значно відрізнятися від однієї системи до іншої; той факт, що все працювало певним чином в конкретному дистрибутиві Linux, не гарантує, що все буде

працювати так само в LFS. LFS має свій власний спосіб роботи, але він дотримується загальноприйнятих стандартів.

Існує альтернативна процедура завантаження, яка називається **systemd**. Ми не будемо далі обговорювати цей процес завантаження. Для детального опису відвідайте <https://www.linux.com/training-tutorials/understanding-and-using-systemd/>.

SysVinit (який відтепер буде називатися «*init*») використовує схему рівнів запуску. Існує сім рівнів запуску, пронумерованих від 0 до 6. (Насправді рівнів запуску більше, але інші призначені для особливих випадків і зазвичай не використовуються. Див. *Init(8)* для отримання детальнішої інформації.) Кожен із семи рівнів відповідає діям, які комп'ютер повинен виконувати під час запуску або вимкнення. Рівень запуску за замовчуванням — 3. Ось опис різних рівнів запуску, як вони реалізовані в LFS:

- 0: зупинити комп'ютер
- 1: режим одного користувача
- 2: зарезервовано для налаштування, в іншому випадку те саме, що й 3
- 3: багатокористувацький режим з мережею
- 4: зарезервовано для налаштування, в іншому випадку те саме, що й 3
- 5: те саме, що й 4, зазвичай використовується для входу в GUI (наприклад, **gdm** у GNOME або **lxdm** у LXDE)
- 6: перезавантажити комп'ютер



Note

Класично, рівень запуску 2 вище визначався як «багатокористувацький режим без мережі», але це було лише багато років тому, коли кілька користувачів могли підключатися до системи через послідовні порти. У сучасному середовищі це не має сенсу, і зараз ми говоримо, що він «зарезервований».

9.6.2. Конфігурація SysVinit

Під час ініціалізації ядра першою програмою, яка запускається (якщо це не перезаписано в командному рядку), є *init*. Ця програма читає файл ініціалізації */etc/inittab*. Створіть цей файл за допомогою:

```

cat > /etc/inittab << "EOF"
# Begin /etc/inittab

id:3:initdefault:
si::sysinit:/etc/rc.d/init.d/rc S
10:0:wait:/etc/rc.d/init.d/rc 0
11:S1:wait:/etc/rc.d/init.d/rc 1
12:2:wait:/etc/rc.d/init.d/rc 2
13:3:wait:/etc/rc.d/init.d/rc 3
14:4:wait:/etc/rc.d/init.d/rc 4
15:5:wait:/etc/rc.d/init.d/rc 5
16:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S06:once:/sbin/sulogin
s1:1:respawn:/sbin/sulogin

1:2345:respawn:/sbin/agetty --noclear tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab
EOF

```

Пояснення цього файлу ініціалізації міститься на сторінці довідки для `inittab`. У LFS ключовою командою є `rc`. Файл ініціалізації вище вказує `rc` запустити всі скрипти, що починаються з `S` у каталозі `/etc/rc.d/rcS.d`, а потім усі скрипти, що починаються з `S` у каталозі `/etc/rc.d/rc?.d`, де знак питання визначається значенням `initdefault`.

Для зручності скрипт `rc` читає бібліотеку функцій у `/lib/lsb/init-functions`. Ця бібліотека також читає опціональний файл конфігурації, `/etc/sysconfig/rc.site`. Будь-які параметри конфігурації системи, описані в наступних розділах, можуть бути розміщені в цьому файлі, що дозволяє об'єднати всі параметри системи в одному файлі.

Для зручності налагодження скрипт функцій також записує весь вивід у файл `/run/var/bootlog`. Оскільки каталог `/run` є `tmpfs`, цей файл не зберігається між завантаженнями; проте він додається до більш постійного файла `/var/log/boot.log` наприкінці процесу завантаження.

9.6.2.1. Зміна рівнів запуску

Зміна рівнів запуску здійснюється за допомогою команди `init <runlevel>`, де `<runlevel>` — це цільовий рівень запуску. Наприклад, щоб перезавантажити комп'ютер, користувач може виконати команду `init 6`, яка є псевдонімом команди `reboot`. Аналогічно, `init 0` є псевдонімом команди `halt`.

У каталозі `/etc/rc.d` є кілька каталогів, які виглядають як `rc?.d` (де ? — номер рівня запуску) та `rcS.d`, і всі вони містять кілька символічних посилань. Деякі посилання починаються з літери K, інші — з літери S, і всі

вони мають дві цифри після початкової літери. Літера K означає зупинку (завершення) служби, а літера S — запуск служби. Цифри визначають порядок виконання скриптів, від 00 до 99 — чим менше число, тим швидше виконується скрипт. Коли **init** переходить на інший рівень запуску, відповідні служби запускаються або зупиняються, залежно від обраного рівня запуску.

Реальні скрипти знаходяться в `/etc/rc.d/init.d`. Вони виконують фактичну роботу, а символільні посилання вказують на них. Посилання K і S вказують на один і той же скрипт в `/etc/rc.d/init.d`. Це пов'язано з тим, що скрипти можуть викликатися з різними параметрами, такими як `start`, `stop`, `restart`, `reload` і `status`. При зустрічі посилання K запускається відповідний скрипт з аргументом `stop`. Коли зустрічається посилання S, відповідний скрипт запускається з аргументом `start`.

Це описи того, що аргументи змушують скрипти робити:

`start`

Служба запущена.

`stop`

Служба зупинена.

`restart`

Служба зупинена, а потім запущена знову.

`reload`

Конфігурація служби оновлена. Використовується після зміни конфігураційного файлу служби, коли службу не потрібно перезапускати.

`status`

Повідомляє, чи працює служба і з якими PID.

Ви можете змінювати спосіб роботи процесу завантаження (адже це ваша власна система LFS). Наведені тут файли є прикладом того, як це можна зробити.

9.6.3. Скрипти завантаження Udev

Скрипт `/etc/rc.d/init.d/udev` запускає **udevd**, активує всі пристрої «coldplug», які вже були створені ядром, і чекає на завершення всіх правил. Скрипт також скасовує обробник `uevent` із стандартного значення `/sbin/hotplug`. Це робиться тому, що ядро більше не потребує виклику зовнішнього бінарного файлу. Натомість **udevd** буде слухати сокет `netlink` на наявність подій `uevent`, які генерує ядро.

Скрипт `/etc/rc.d/init.d/udev_retry` відповідає за повторне запускання подій для підсистем, правила яких можуть залежати від файлових систем, що не змонтовані до запуску скрипта **mountfs** (зокрема, це може стосуватися `/usr` та `/var`). Цей скрипт запускається після скрипта **mountfs**, тому ці правила (якщо вони запускаються повторно) повинні спрацювати з другої спроби. Він налаштовується файлом `/etc/sysconfig/udev_retry`; будь-які слова в цьому файлі, крім коментарів, вважаються іменами підсистем, які спрацьовують під час повторного запуску. Щоб знайти підсистему пристрою, використовуйте **udevadm info --attribute-walk <device>**, де `<device>` — це абсолютний шлях у `/dev` або `/sys`, наприклад `/dev/sr0` або `/sys/class/rtc`.

Інформацію про завантаження модулів ядра та udev див. у розділі 9.3.2.3, «Завантаження модулів».

9.6.4. Налаштування системного годинника

Скрипт **setclock** читає час з апаратного годинника, також відомого як BIOS або Complementary Metal Oxide Semiconductor (CMOS) clock. Якщо апаратний годинник налаштований на UTC, цей скрипт перетворює час апаратного годинника на місцевий час, використовуючи файл `/etc/localtime` (який вказує програмі **hwclock**, який часовий пояс використовувати). Немає способу визначити, чи встановлений апаратний годинник на UTC, тому це потрібно налаштувати вручну.

Програма **setclock** запускається через udev, коли ядро виявляє апаратні можливості під час завантаження. Її також можна запустити вручну з параметром `stop`, щоб зберегти системний час у годиннику CMOS.

Якщо ви не пам'ятаєте, чи встановлений апаратний годинник на UTC, дізнайтесь це, запустивши команду **hwclock -localtime --show**. Це відобразить поточний час відповідно до апаратного годинника. Якщо цей час збігається з тим, що показує ваш годинник, то апаратний годинник встановлений на місцевий час. Якщо вихідні дані **hwclock** не відповідають місцевому часу, ймовірно, він встановлений на час UTC. Перевірте це, додавши або віднявши відповідну кількість годин для вашого часового поясу до часу, показаного **hwclock**. Наприклад, якщо ви зараз перебуваєте в часовому поясі MST, який також відомий як GMT-0700, додайте сім годин до місцевого часу.

Змініть значення змінної `UTC` нижче на 0 (нуль), якщо апаратний годинник НЕ налаштований на час UTC.

Створіть новий файл `/etc/sysconfig/clock`, виконавши наступне:

```
cat > /etc/sysconfig/clock << "EOF"
# Begin /etc/sysconfig/clock

UTC=1

# Set this to any options you might need to give to hwclock,
# such as machine hardware clock type for Alphas.
CLOCKPARAMS=

# End /etc/sysconfig/clock
EOF
```

Хороша підказка, що пояснює, як працювати з часом в LFS, доступна за адресою <https://www.linuxfromscratch.org/hints/downloads/files/time.txt>. Вона пояснює такі питання, як часові пояси, UTC та змінна середовища TZ.



Note

Параметри `CLOCKPARAMS` та `UTC` також можна встановити у файлі `/etc/sysconfig/rc.site`.

9.6.5. Налаштування консолі Linux

У цьому розділі розглядається, як налаштувати скрипт завантаження **console**, який встановлює розкладку клавіатури, шрифт консолі та рівень журналу ядра консолі. Якщо символи, що не належать до ASCII (наприклад, знак авторського права, знак британського фунта та символ євро) не будуть використовуватися, а клавіатура є американською, більшу частину цього розділу можна пропустити. Без конфігураційного

файлу (або еквівалентних налаштувань у `rc.site`) скрипт завантаження **console** не буде виконувати жодних дій.

Скрипт **console** читує файл `/etc/sysconfig/console` для отримання інформації про конфігурацію. Визначте, яка клавіатурна розкладка та шрифт екрану будуть використовуватися. Різні HOWTO для конкретних мов також можуть допомогти в цьому; див. <https://tldp.org/HOWTO/HOWTO-INDEX/other-lang.html>. Якщо ви все ще маєте сумніви, пошукайте в каталогах `/usr/share/keystrokes` та `/usr/share/consolefonts` дійсні розкладки клавіатури та шрифти екрану. Прочитайте сторінки довідки *loadkeys(1)* та *setfont(8)*, щоб визначити правильні аргументи для цих програм.

Файл `/etc/sysconfig/console` повинен містити рядки у форматі: змінна=значення. Розпізнаються такі змінні:

LOGLEVEL

Ця змінна визначає рівень журналу для повідомлень ядра, що надсилаються до консолі, як встановлено **dmesg -n**. Допустимі рівні — від 1 (без повідомлень) до 8. Рівень за замовчуванням — 7, що є досить докладним.

KEYMAP

Ця змінна визначає аргументи для програми **loadkeys**, зазвичай ім'я клавіатурної розкладки, що завантажується, наприклад, `it`. Якщо ця змінна не встановлена, скрипт завантаження не запустить програму **loadkeys**, і буде використана клавіатурна розкладка ядра за замовчуванням. Зверніть увагу, що деякі клавіатурні розкладки мають кілька версій з однаковою назвою (`cz` та її варіанти в `qwerty/` та `qwertz/`, `es` в `olpc/` та `qwerty/`, а також `trf` в `fgGlob/` та `qwerty/`). У таких випадках також слід вказати батьківський каталог (наприклад, `qwerty/es`), щоб забезпечити завантаження правильної клавіатурної розкладки.

KEYMAP_CORRECTIONS

Ця (рідко використовувана) змінна визначає аргументи для другого виклику програми **loadkeys**. Це корисно, якщо стандартна розкладка клавіатури не є повністю задовільною і потрібно внести невеликі корективи. Наприклад, щоб включити знак євро в розкладку клавіатури, яка зазвичай його не має, встановіть для цієї змінної значення `euro2`.

FONT

Ця змінна визначає аргументи для програми **setfont**. Зазвичай це включає ім'я шрифту, `-m` та ім'я карти символів програми, яку потрібно завантажити. Наприклад, щоб завантажити шрифт «`lat1-16`» разом із картою символів програми «`8859-1`» (відповідно для США), встановіть для цієї змінної значення `lat1-16 -m 8859-1`. У режимі UTF-8 ядро використовує карту символів програми для перетворення 8-бітних кодів клавіш у UTF-8. Тому аргумент параметра «`-m`» повинен бути встановлений на кодування складених кодів клавіш у карті клавіш.

UNICODE

Встановіть цю змінну на `1`, `yes` або `true`, щоб перевести консоль у режим UTF-8. Це корисно в локалях на основі UTF-8, а в інших випадках шкідливо.

LEGACY_CHARSET

Для багатьох розкладок клавіатури в пакеті `Kbd` немає стандартної Unicode-карти клавіш. Скрипт завантаження **console** перетворить доступну карту клавіш в UTF-8 на льоту, якщо ця змінна встановлена на кодування доступної не-UTF-8 карти клавіш.

Деякі приклади:

- Ми будемо використовувати `C.UTF-8` як локаль для інтерактивних сесій у консолі Linux у розділі 9.7, «Налаштування системної локалі», тому ми повинні встановити `UNICODE` на 1. А шрифти консолі, що постачаються пакетом `Kbd` і містять гліфи для всіх символів з повідомлень програми в локалі `C.UTF-8`, — це `LatArCyrHeb*.psfu.gz`, `LatGrkCyr*.psfu.gz`, `Lat2-Terminus16.psfu.gz` та `pancyrillic.f16.psfu.gz` у `/usr/share/consolefonts` (інші шрифти консолі, що постачаються, не мають гліфів деяких символів, таких як ліва/права лапки Unicode та англійський тире Unicode). Тому встановіть один із них, наприклад `Lat2-Terminus16.psfu.gz`, як шрифт консолі за замовчуванням:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
FONT="Lat2-Terminus16"

# End /etc/sysconfig/console
EOF
```

- Для налаштування, що не використовує Unicode, зазвичай потрібні лише змінні `KEYMAP` та `FONT`. Наприклад, для польського налаштування використовується:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# End /etc/sysconfig/console
EOF
```

- Як згадувалося вище, іноді необхідно дещо скоригувати стандартну розкладку клавіатури. У наведеному нижче прикладі до німецької розкладки клавіатури додається символ євро:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"
UNICODE="1"

# End /etc/sysconfig/console
EOF
```

- Нижче наведено приклад з підтримкою Unicode для болгарської мови, для якої існує стандартна клавіатурна розкладка UTF-8:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
```

```
FONT="LatArCyrHeb-16"
```

```
# End /etc/sysconfig/console
EOF
```

- Через використання шрифту LatArCyrHeb-16 з 512 символами у попередньому прикладі яскраві кольори більше не доступні на консолі Linux, якщо не використовується фреймбуфер. Якщо ви хочете мати яскраві кольори без фреймбуфера і можете обйтися без символів, що не належать до вашої мови, ви все ще можете використовувати мовно-спеціфічний шрифт з 256 символами, як показано нижче:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console
```

```
UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"
```

```
# End /etc/sysconfig/console
EOF
```

- Наступний приклад ілюструє автоматичне перетворення клавіатурної розкладки з ISO-8859-15 на UTF-8 та увімкнення мертвих клавіш у режимі Unicode:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console
```

```
UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
FONT="LatArCyrHeb-16 -m 8859-15"
```

```
# End /etc/sysconfig/console
EOF
```

- Деякі клавіатурні розкладки мають мертві клавіші (тобто клавіші, які самі по собі не створюють символів, але ставлять акцент на символі, створеному наступною клавішою) або визначають правила комбінування (наприклад: «натисніть Ctrl+. A E, щоб отримати Є» у стандартній клавіатурній розкладці). Linux-6.13.4 правильно інтерпретує мертві клавіші та правила комбінування в клавіатурній розкладці лише тоді, коли символи, що комбінуються, не є багатобайтовими. Цей недолік не впливає на клавіатурні розкладки для європейських мов, оскільки в них акценти додаються до неакцентованих символів ASCII або комбінуються два символи ASCII. Однак у режимі UTF-8 це є проблемою; наприклад, для грецької мови, де іноді потрібно поставити акцент на літері α. Рішенням є або уникнення використання UTF-8, або встановлення системи X window, яка не має цього обмеження в обробці вводу.
- Для китайської, японської, корейської та деяких інших мов консоль Linux не може бути налаштована для відображення необхідних символів. Користувачі, яким потрібні такі мови, повинні встановити систему X Window, шрифти, що охоплюють необхідні діапазони символів, та відповідний метод введення (наприклад, SCIM підтримує широкий спектр мов).

**Note**

Файл `/etc/sysconfig/console` контролює лише локалізацію текстової консолі Linux. Він не має нічого спільного з налаштуванням відповідної розкладки клавіатури та шрифтів терміналу в системі X Window, сеансами ssh або послідовною консолью. У таких ситуаціях обмеження, згадані в останніх двох пунктах списку вище, не застосовуються.

9.6.6. Створення файлів під час завантаження

Іноді бажано створювати файли під час завантаження системи. Наприклад, часто потрібна директорія `/tmp/.ICE-unix`. Це можна зробити, створивши запис у конфігураційному скрипті `/etc/sysconfig/createfiles`. Формат цього файла вказано в коментарях до файлу конфігурації за замовчуванням.

9.6.7. Налаштування скрипта Sysklogd

Скрипт `sysklogd` викликає програму `syslogd` як частину ініціалізації System V. Опція `-m 0` вимикає періодичну позначку часу, яку `syslogd` за замовчуванням записує в файли журналу кожні 20 хвилин. Якщо ви хочете увімкнути цю періодичну позначку часу, відредактуйте файл `/etc/sysconfig/rc.site` і задайте змінний `SYSKLOGD_PARMS` бажане значення. Наприклад, щоб видалити всі параметри, задайте змінний нульове значення:

```
SYSKLOGD_PARMS=
```

Більше опцій дивіться у `man syslogd`.

9.6.8. Файл rc.site

Додатковий файл `/etc/sysconfig/rc.site` містить налаштування, які автоматично встановлюються для кожного скрипта завантаження SystemV. Він може також встановлювати значення, вказані у файлах `hostname`, `console` та `clock` у каталозі `/etc/sysconfig/`. Якщо відповідні змінні присутні як у цих окремих файлах, так і в `rc.site`, то значення у файлах, що стосуються конкретного скрипта, беруть верх.

`rc.site` також містить параметри, які дозволяють налаштовувати інші аспекти процесу завантаження. Встановлення змінної `IPROMPT` дозволить вибірково запускати скрипти завантаження. Інші опції описані в коментарях до файла. Стандартна версія файла виглядає наступним чином:

```
# rc.site
# Optional parameters for boot scripts.

# Distro Information
# These values, if specified here, override the defaults
#DISTRO="Linux From Scratch" # The distro name
#DISTRO_CONTACT="lfs-dev@lists.linuxfromscratch.org" # Bug report address
#DISTRO_MINI="LFS" # Short name used in filenames for distro config

# Define custom colors used in messages printed to the screen

# Please consult `man console_codes` for more information
# under the "ECMA-48 Set Graphics Rendition" section
```

```

#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font. This does
# not affect framebuffer consoles

# These values, if specified here, override the defaults
#BRACKET="\\"033[1;34m"      # Blue
#FAILURE="\\"033[1;31m"      # Red
#INFO="\\"033[1;36m"         # Cyan
#NORMAL="\\"033[0;39m"        # Grey
#SUCCESS="\\"033[1;32m"       # Green
#WARNING="\\"033[1;33m"       # Yellow

# Use a colored prefix
# These values, if specified here, override the defaults
#BMPREFIX=""
#SUCCESS_PREFIX="${SUCCESS} * ${NORMAL} "
#FAILURE_PREFIX="${FAILURE}*****${NORMAL} "
#WARNING_PREFIX="${WARNING} *** ${NORMAL} "

# Manually set the right edge of message output (characters)
# Useful when resetting console font during boot to override
# automatic screen width detection
#COLUMNS=120

# Interactive startup
#IPROMPT="yes"    # Whether to display the interactive boot prompt
#itime="3"          # The amount of time (in seconds) to display the prompt

# The total length of the distro welcome string, without escape codes
#wlen=$(echo "Welcome to ${DISTRO}" | wc -c )
#welcome_message="Welcome to ${INFO}${DISTRO}${NORMAL}"

# The total length of the interactive string, without escape codes
#ilen=$(echo "Press 'I' to enter interactive startup" | wc -c )
#i_message="Press '${FAILURE}I${NORMAL}' to enter interactive startup"

# Set scripts to skip the file system check on reboot
#FASTBOOT=yes

# Skip reading from the console
#HEADLESS=yes

# Write out fsck progress if yes
#VERBOSE_FSCK=no

# Speed up boot without waiting for settle in udev
#OMIT_UDEV_SETTLE=y

```

```
# Speed up boot without waiting for settle in udev_retry
#OMIT_UDEV_RETRY_SETTLE=yes

# Skip cleaning /tmp if yes
#SKIPTMPCLEAN=no

# For setclock
#UTC=1
#CLOCKPARAMS=

# For consolelog (Note that the default, 7=debug, is noisy)
#LOGLEVEL=7

# For network
#HOSTNAME=mylfs

# Delay between TERM and KILL signals at shutdown
#KILLDELAY=3
# Optional sysklogd parameters
#SYSKLOGD_PARMs="-m 0"

# Console parameters
#UNICODE=1
#KEYMAP="de-latin1"
#KEYMAP_CORRECTIONS="euro2"
#FONT="lat0-16 -m 8859-15"
#LEGACY_CHARSET=
```

9.6.8.1. Налаштування скриптів завантаження та вимкнення

Скрипти завантаження LFS завантажують і вимикають систему досить ефективно, але є кілька налаштувань, які ви можете зробити у файлі `rc.site`, щоб ще більше підвищити швидкість і налаштувати повідомлення відповідно до ваших уподобань. Для цього налаштуйте параметри у файлі `/etc/sysconfig/rc.site` вище.

- Під час виконання скрипта завантаження `udev` відбувається виклик **`udev settle`**, який вимагає деякого часу для виконання. Цей час може бути необхідним або не необхідним, залежно від пристрійв у системі. Якщо у вас є лише прості розділи та одна мережева карта, процес завантаження, ймовірно, не потребуватиме очікування на виконання цієї команди. Щоб пропустити її, встановіть змінну `OMIT_UDEV_SETTLE=y`.
- Скрипт завантаження `udev_retry` також запускає **`udev settle`** за замовчуванням. Ця команда потрібна тільки в тому випадку, якщо каталог `/var` монтирується окремо, оскільки годиннику потрібен файл `/var/lib/hwclock/adjtime`. Інші налаштування також можуть потребувати очікування завершення роботи `udev`, але в багатьох установках це є необхідним. Пропустіть команду, встановивши змінну `OMIT_UDEV_RETRY_SETTLE=y`.
- За замовчуванням перевірка файлової системи виконується без виведення повідомлень. Це може виглядати як затримка під час завантаження системи. Щоб увімкнути виведення повідомлень `fsck`, встановіть змінну `VERBOSE_FSCK=y`.

- Під час перезавантаження ви можете повністю пропустити перевірку файлової системи **fsck**. Для цього створіть файл `/fastboot` або перезавантажте систему за допомогою команди `/sbin/shutdown -f -r now`. З іншого боку, ви можете змусити перевірити всі файлові системи, створивши файл `/forcefsck` або запустивши **shutdown** з параметром `-F` замість `-f`.

Встановлення змінної `FASTBOOT=y` вимкне **fsck** під час процесу завантаження, доки вона не буде видалена. Це не рекомендується робити на постійній основі.

- Зазвичай усі файли в каталозі `/tmp` видаляються під час завантаження системи. Залежно від кількості файлів або каталогів, що містяться в ньому, це може спричинити помітне уповільнення процесу завантаження. Щоб пропустити видалення цих файлів, встановіть змінну `SKIPTMPCLEAN=y`.
- Під час вимкнення програма **init** надсилає сигнал `TERM` до кожної програми, яку вона запустила (наприклад, `agetty`), чекає встановлений час (за замовчуванням 3 секунди), потім надсилає кожному процесу сигнал `KILL` і знову чекає. Цей процес повторюється в скрипті **sendsignals** для всіх процесів, які не вимикаються власними скриптами. Затримку для **init** можна встановити за допомогою параметра. Наприклад, щоб усунути затримку в **init**, передайте параметр `-t0` під час вимкнення або перезавантаження (наприклад, `/sbin/shutdown -t0 -r now`). Затримку для скрипта **sendsignals** можна пропустити, встановивши параметр `KILLDELAY=0`.

9.7. Налаштування локальних параметрів системи

Деякі змінні середовища необхідні для підтримки рідної мови. Їх правильне налаштування надає:

- Виведення програм, що перекладаються на вашу рідну мову.
- Правильна класифікація символів на літери, цифри та інші класи. Це необхідно для того, щоб **bash** правильно приймав не-ASCII символи в командних рядках в неанглійських локалях.
- Правильний алфавітний порядок сортування для країни.
- Відповідний розмір паперу за замовчуванням.
- Правильне форматування грошових, часових та датових значень.

Замініть `<11>` нижче дволітерним кодом бажаної мови (наприклад, `en`) та `<CC>` дволітерним кодом відповідної країни (наприклад, `GB`). `<charmap>` слід замінити канонічним `charmap` для обраної локалі. Також можуть бути присутні опціональні модифікатори, такі як `@euro`.

Список усіх локалей, що підтримуються Glibc, можна отримати, виконавши таку команду:

```
locale -a
```

Charmaps можуть мати кілька псевдонімів, наприклад, `ISO-8859-1` також називається `iso8859-1` та `iso88591`. Деякі програми не можуть правильно обробляти різні синоніми (наприклад, вимагають, щоб `UTF-8` писалося як `UTF-8`, а не `utf8`), тому у більшості випадків найбезпечніше вибирати канонічне ім'я для конкретної локалі. Щоб визначити канонічне ім'я, виконайте наступну команду, де `<ім'я локалі>` — це результат, отриманий за допомогою команди **locale -a** для вашої бажаної локалі (`en_GB.iso88591` у нашому прикладі).

```
LC_ALL=<locale name> locale charmap
```

Для локалі `en_GB.iso88591` вищевказана команда виведе:

```
ISO-8859-1
```

В результаті отримуємо остаточне налаштування локалі `en_GB.ISO-8859-1`. Важливо, щоб локаль, знайдена за допомогою вищезазначененої евристики, була перевірена перед додаванням до файлів запуску Bash:

```
LC_ALL=<locale name> locale language
LC_ALL=<locale name> locale charmap
LC_ALL=<locale name> locale int_curr_symbol
LC_ALL=<locale name> locale int_prefix
```

Вищезазначені команди повинні вивести назву мови, кодування символів, яке використовується локаллю, місцеву валюту, а також префікс, який потрібно набрати перед телефонним номером, щоб зв'язатися з цією країною. Якщо будь-яка з вищезазначених команд не працює і виводить повідомлення, подібне до наведеного нижче, це означає, що ваша локаль або не була встановлена в розділі 8, або не підтримується стандартною інсталяцією Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Якщо це трапиться, вам слід або встановити потрібну локаль за допомогою команди `localeddef`, або розглянути можливість вибору іншої локалі. Далі в інструкціях передбачається, що таких повідомлень про помилки від Glibc не буде.

Інші пакети також можуть працювати некоректно (але не обов'язково відображати повідомлення про помилки), якщо назва локалі не відповідає їхнім очікуванням. У таких випадках вивчення того, як інші дистрибутиви Linux підтримують вашу локаль, може надати корисну інформацію.

Програма оболонки `/bin/bash` (далі «оболонка») використовує набір файлів запуску для створення середовища для роботи. Кожен файл має конкретне призначення і може по-різному впливати на середовище входу в систему та інтерактивне середовище. Файли в каталозі `/etc` містять глобальні налаштування. Якщо в домашньому каталозі є еквівалентні файли, вони можуть замінити глобальні налаштування.

Інтерактивна оболонка входу запускається після успішного входу за допомогою `/bin/login`, зчитуючи файл `/etc/passwd`. Інтерактивна оболонка без входу запускається в командному рядку (наприклад, `[prompt] $/bin/bash`). Неінтерактивна оболонка зазвичай присутня під час виконання скрипта оболонки. Вона є неінтерактивною, оскільки обробляє скрипт і не чекає на введення користувача між командами.

Створіть файл `/etc/profile` після визначення відповідних налаштувань локалі, щоб встановити бажану локаль, але встановіть локаль `C.UTF-8`, якщо ви працюєте в консолі Linux (щоб запобігти виведенню програм символів, які консоль Linux не може відобразити):

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

for i in $(locale); do
    unset ${i%=*}
done

if [[ "$TERM" = linux ]]; then
    export LANG=C.UTF-8
else
    export LANG=<11>_<CC>.<charmap><@modifiers>
fi

# End /etc/profile
EOF
```

Локалі `c` (за замовчуванням) та `en_US` (рекомендована для користувачів англійської мови США) відрізняються. С використовує 7-бітний набір символів US-ASCII і розглядає байти з встановленим старшим бітом як недійсні символи. Ось чому, наприклад, команда `ls` замінює їх знаками питання в цій локалі. Крім того, спроба надіслати пошту з такими символами з Mutt або Pine призводить до надсилення повідомлень, що не відповідають RFC (набір символів у вихідній пошті вказується як `unknown 8-bit`). Рекомендується використовувати локаль `c` тільки в тому випадку, якщо ви впевнені, що вам ніколи не знадобляться 8-бітні символи.

9.8. Створення файлу /etc/inputrc

Файл `inputrc` є файлом конфігурації для бібліотеки `readline`, яка надає можливості редагування під час введення користувачем рядка з терміналу. Він працює шляхом перетворення вхідних даних з клавіатури в конкретні дії. `Readline` використовується `bash` та більшістю інших оболонок, а також багатьма іншими програмами.

Більшість людей не потребують функціональності, специфічної для користувача, тому команда нижче створює глобальний файл `/etc/inputrc`, який використовується всіма, хто входить у систему. Якщо пізніше ви вирішите, що вам потрібно замінити стандартні налаштування для кожного користувача окремо, ви можете створити файл `.inputrc` у домашньому каталозі користувача зі зміненими відповідностями.

Для отримання додаткової інформації про те, як редагувати файл `inputrc`, див. `info bash` у розділі *Readline Init File*. `info readline` також є хорошим джерелом інформації.

Нижче наведено загальний глобальний файл `inputrc` разом із коментарями, що пояснюють призначення різних опцій. Зверніть увагу, що коментарі не можуть бути в тому самому рядку, що й команди. Створіть файл за допомогою такої команди:

```

cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8-bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions
"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF

```

9.8. Створення файлу /etc/shells

Файл `shells` містить список оболонок входу в систему. Програми використовують цей файл для визначення того, чи є оболонка дійсна. Дляожної оболонки повинен бути присутній один рядок, що складається з шляху до оболонки відносно кореня структури каталогів (/).

Наприклад, цей файл використовується командою **chsh** для визначення, чи може користувач без привілеїв змінити оболонку входу для свого облікового запису. Якщо ім'я команди не вказано в списку, користувачеві буде відмовлено в можливості змінити оболонку.

Це вимога для таких програм, як GDM, які не заповнюють браузер облич, якщо не можуть знайти /etc/shells, або FTP-демони, які традиційно забороняють доступ користувачам із оболонками, що не включені до цього файлу.

```
cat > /etc/shells << "EOF"
# Begin /etc/shells

/bin/sh
/bin/bash

# End /etc/shells
EOF
```

Глава 10. Створення завантажувальної системи LFS

10.1. Вступ

Настав час зробити систему LFS завантажувальною. У цьому розділі розглядається створення файлу /etc/fstab, компіляція ядра для нової системи LFS та встановлення завантажувача GRUB, щоб систему LFS можна було вибрати для завантаження під час запуску.

10.2. Створення файлу /etc/fstab

Файл /etc/fstab використовується деякими програмами для визначення, де файлові системи мають бути змонтовані за замовчуванням, в якому порядку та які з них необхідно перевірити (на наявність помилок цілісності) перед монтуванням. Створіть нову таблицю файлових систем таким чином:

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab
# file system    mount-point      type        options          dump   fsck
#                                         order

/dev/<xxx>      /           <fff>      defaults        1      1
/dev/<yyy>      swap        swap        pri=1          0      0
proc            /proc        proc        nosuid,noexec,nodev 0      0
sysfs           /sys         sysfs       nosuid,noexec,nodev 0      0
devpts           /dev/pts    devpts      gid=5,mode=620    0      0
tmpfs            /run         tmpfs       defaults        0      0
devtmpfs         /dev         devtmpfs   mode=0755,nosuid 0      0
tmpfs             /dev/shm    tmpfs       nosuid,nodev      0      0
cgroup2          /sys/fs/cgroup cgroup2    nosuid,noexec,nodev 0      0

# End /etc/fstab
EOF
```

Замініть <xxx>, <yyy> та <fff> значеннями, що відповідають системі, наприклад, sda2, sda5 та ext4. Детальніше про шість полів у цьому файлі див. *fstab(5)*.

Файлові системи з походженням MS-DOS або Windows (тобто vfat, ntfs, smbfs, cifs, iso9660, udf) потребують спеціальної опції utf8, щоб символи, що не належать до ASCII, у назвах файлів інтерпретувалися правильно. Для локалей, що не є UTF-8, значення iocharset повинно бути встановлено таким самим, як і набір символів локалі, скоригований таким чином, щоб ядро його розуміло. Це працює, якщо відповідне визначення набору символів (знаходиться в розділі «Файлові системи» -> «Підтримка рідної мови» під час налаштування ядра) було скомпільовано в ядро або побудовано як модуль. Однак, якщо набір символів локалі є UTF-8, відповідна опція iocharset=utf8 зробить файлову систему чутливою до реєстру. Щоб виправити це, використовуйте спеціальну опцію utf8 замість iocharset=utf8 для локалей UTF-8. Опція «codepage» також потрібна для файлових систем vfat та smbfs. Вона повинна бути встановлена на номер кодової сторінки, що використовується в MS-DOS у вашій країні. Наприклад, для монтування USB-флеш-накопичувачів користувачеві ru_RU.KOI8-R потрібно буде вказати наступне в частині опцій рядка монтування в /etc/fstab:

```
noauto,user,quiet,showexec,codepage=866,iocharset=koi8r
```

Відповідний фрагмент опцій для користувачів ru_RU.UTF-8:

```
noauto,user,quiet,showexec,codepage=866,utf8
```

Зверніть увагу, що використання `iocharset` є стандартним для `iso8859-1` (що зберігає файлову систему нечутливою до регістру), а опція `utf8` повідомляє ядру перетворити імена файлів за допомогою UTF-8, щоб їх можна було інтерпретувати в локалі UTF-8.

Також можна вказати значення кодової сторінки та набору символів за замовчуванням для деяких файлових систем під час конфігурації ядра. Відповідні параметри мають назви «Опція NLS за замовчуванням» (`CONFIG_NLS_DEFAULT`), «Опція віддаленого NLS за замовчуванням» (`CONFIG_SMB_NLS_DEFAULT`), «Default codepage for FAT» (`CONFIG_FAT_DEFAULT_CODEPAGE`) та «Default iocharset for FAT» (`CONFIG_FAT_DEFAULT_IOSCHARSET`). Немає можливості вказати ці налаштування для файлової системи `ntfs` під час компіляції ядра.

Для деяких типів жорстких дисків можна зробити файлову систему `ext3` надійною у разі збою живлення. Для цього додайте опцію монтування `barrier=1` до відповідного запису в файлі `/etc/fstab`. Щоб перевірити, чи підтримує дисковий привід цю опцію, запустіть `hdparm` на відповідному дисковому приводі. Наприклад, якщо:

```
hdparm -I /dev/sda | grep NCQ
```

повертає не порожній вивід, опція підтримується.

Примітка: розділи на основі управління логічними томами (LVM) не можуть використовувати опцію `barrier`.

10.3. Linux-6.13.4

Пакет Linux містить ядро Linux.

Приблизний час побудови: 0.4 – 32 SBU (зазвичай близько 2.5 SBU)

Необхідний простір на диску: 1.7 - 14 GB (зазвичай близько 2.3 GB)

10.3.1. Інсталляція ядра

Створення ядра включає кілька етапів — конфігурацію, компіляцію та інсталляцію. Прочитайте файл README у дереві джерел ядра, щоб дізнатися про альтернативні методи конфігурації ядра, на відміну від тих, що описані в цій книзі.



Important

Створення ядра Linux вперше є одним з найскладніших завдань у LFS. Правильне виконання цього завдання залежить від конкретного обладнання цільової системи та ваших конкретних потреб. Існує майже 12 000 елементів конфігурації, доступних для ядра, хоча для більшості комп'ютерів потрібна лише третина з них. Редактори LFS рекомендують користувачам, які не знайомі з цим процесом, досить ретельно дотримуватися наведених нижче процедур. Мета полягає в тому, щоб отримати початкову систему, в якій ви зможете увійти в систему з командного рядка після перезавантаження в розділі 11.3, «Перезавантаження системи». На цьому етапі оптимізація та налаштування не є метою.

Загальну інформацію про конфігурацію ядра див.
<https://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt>. Додаткову інформацію про конфігурацію та побудову ядра можна знайти за адресою <https://anduin.linuxfromscratch.org/LFS/kernel-nutshell/>. Ці посилання дещо застарілі, але все ще дають розумний огляд процесу.

Якщо все інше не допоможе, ви можете звернутися за допомогою до списку розсилки *lfs-support*. Зверніть увагу, що для цього необхідно підписатися на список, щоб уникнути спаму.

Підготуйтесь до компіляцію, виконавши наступну команду:

make mrproper

Це гарантує, що дерево ядра є абсолютно чистим. Команда розробників ядра рекомендує виконувати цю команду перед кожною компіляцією ядра. Не покладайтесь на те, що дерево джерел буде чистим після розпакування.

Існує кілька способів налаштування опцій ядра. Зазвичай це робиться за допомогою меню, наприклад:

make menuconfig

Значення опціональних змінних середовища make:

LANG=<host_LANG_value> LC_ALL=

Це встановлює локальні налаштування, які використовуються на хості. Це може бути потрібно для правильного відображення інтерфейсу menuconfig ncurses на текстовій консолі Linux з кодуванням UTF-8. Якщо використовується, обов'язково замініть <host_LANG_value> на значення змінної \$LANG з вашого хосту. Ви можете замість цього використовувати значення \$LC_ALL або \$LC_CTYPE хосту.

make menuconfig

Це запускає інтерфейс ncurses, керований меню. Для інших (графічних) інтерфейсів введіть **make help**.



Note

Хорошим початком для налаштування конфігурації ядра є запуск команди **make defconfig**. Це встановить базову конфігурацію в оптимальний стан, що враховує архітектуру вашої поточної системи.

Обов'язково увімкніть/вимкніть/налаштуйте наступні функції, інакше система може працювати некоректно або взагалі не завантажуватися:

```

General setup --->
  [ ] Compile the kernel with warnings as errors                               [WERROR]
  CPU/Task time and stats accounting --->
    [*] Pressure stall information tracking                                     [PSI]
    [ ]   Require boot parameter to enable pressure stall information tracking
          ... [PSI_DEFAULT_DISABLED]
  < > Enable kernel headers through /sys/kernel/kheaders.tar.xz      [IKHEADERS]
  [*] Control Group support --->                                              [CGROUPS]
    [*] Memory controller                                                       [MEMCG]
  [ ] Configure standard kernel features (expert users) --->             [EXPERT]

Processor type and features --->
  [*] Build a relocatable kernel                                               [RELOCATABLE]
  [*] Randomize the address of the kernel image (KASLR)                         [RANDOMIZE_BASE]

General architecture-dependent options --->
  [*] Stack Protector buffer overflow detection                                [STACKPROTECTOR]
  [*] Strong Stack Protector                                                 [STACKPROTECTOR_STRONG]

Device Drivers --->
  Generic Driver Options --->
    [ ] Support for uevent helper                                              [UEVENT_HELPER]
    [*] Maintain a devtmpfs filesystem to mount at /dev                         [DEVTMPFS]
    [*] Automount devtmpfs at /dev, after the kernel mounted the rootfs
        ... [DEVTMPFS_MOUNT]

Firmware Drivers --->
  [*] Mark VGA/VBE/EFI FB as generic system framebuffer                         [SYSFB_SIMPLEFB]
  Graphics support --->
    <> Direct Rendering Manager (XFree86 4.1.0 and higher DRI support) --->
        ... [DRM]
    [*] Display a user-friendly message when a kernel panic occurs
        ... [DRM_PANIC]
  (kmsg) Panic screen formatter                                                 [DRM_PANIC_SCREEN]
  Supported DRM clients --->
    [*] Enable legacy fbdev support for your modesetting driver
        ... [DRM_FBDEV_EMULATION]
    <> Simple framebuffer driver                                                [DRM_SIMPLEDRM]
  Console display driver support --->
    [*] Framebuffer Console support                                              [FRAMEBUFFER_CONSOLE]

```

Увімкніть деякі додаткові функції, якщо ви створюєте 64-бітну систему. Якщо ви використовуєте menuconfig, увімкніть їх у такому порядку: спочатку CONFIG_PCI_MSI, потім CONFIG_IRQ_REMAP, і наостанок CONFIG_X86_X2APIC, оскільки опція з'являється тільки після вибору її залежностей.

```

Processor type and features --->
  [*] Support x2apic [X86_X2APIC]

Device Drivers --->
  [*] PCI support ---> [PCI]
    [*] Message Signaled Interrupts (MSI and MSI-X) [PCI_MSI]
    [*] IOMMU Hardware Support ---> [IOMMU_SUPPORT]
      [*] Support for Interrupt Remapping [IRQ_REMAP]

```

Якщо ви створюєте 32-роздрядну систему, що працює на апаратному забезпеченні з об'ємом оперативної пам'яті більше 4 ГБ, налаштуйте конфігурацію так, щоб ядро могло використовувати до 64 ГБ фізичної оперативної пам'яті:

```

Processor type and features --->
  High Memory Support ---> [HIGHMEM64G]
    (X) 64GB

```

Якщо розділ для системи LFS знаходиться на SSD NVME (тобто вузол пристрою для розділу є `/dev/nvme*` замість `/dev/sd*`), увімкніть підтримку NVME, інакше система LFS не завантажиться:

```

PDevice Drivers --->
  NVME Support ---> [BLK_DEV_NVME]
    <*> NVM Express block device

```

Існує кілька інших опцій, які можуть бути потрібні залежно від вимог до системи. Список опцій, необхідних



Note

Якщо ваше хост-обладнання використовує UEFI і ви хочете завантажити систему LFS за допомогою нього, вам слід налаштувати деякі параметри ядра відповідно до сторінки BLFS, **навіть якщо ви будете використовувати завантажувач UEFI з хост-дистрибутива.**

Обґрунтування вищезазначених елементів конфігурації:

Randomize the address of the kernel image (KASLR)

Увімкніти ASLR для образу ядра, щоб зменшити ризик деяких атак, заснованих на фікованих адресах конфіденційних даних або коду в ядрі.

Compile the kernel with warnings as errors

Це може призвести до збою компіляції, якщо компілятор та/або конфігурація відрізняються від тих, що використовують розробники ядра.

Enable kernel headers through /sys/kernel/kheaders.tar.xz

Для цього потрібно, щоб **сріо** компілював ядро. **сріо** не встановлюється LFS.

Configure standard kernel features (expert users)

Це призведе до появи деяких опцій в інтерфейсі конфігурації, але зміна цих опцій може бути небезпечною. Не використовуйте цю опцію, якщо не знаєте, що робите.

Strong Stack Protector

Увімкніть SSP для ядра. Ми увімкнули його для всього простору користувача за допомогою `--enable-default-ssp` під час конфігурації GCC, але ядро не використовує стандартні налаштування GCC для SSP. Ми увімкнемо його явно тут.

Support for uevent helper

Встановлення цієї опції може заважати управлінню пристроями при використанні Udev.

Maintain a devtmpfs

Це створить автоматизовані вузли пристройів, які заповнюються ядром, навіть без запуску Udev. Udev потім працює поверх цього, керуючи дозволами та додаючи символічні посилання. Цей елемент конфігурації необхідний для всіх користувачів Udev.

Automount devtmpfs at /dev

Це монтує ядро пристройів в `/dev` при переході до кореневої файлової системи безпосередньо перед запуском `init`.

Display a user-friendly message when a kernel panic occurs

Це змусить ядро правильно відображати повідомлення у випадку паніки ядра та якщо запущений драйвер DRM підтримує таку функцію. Без цього діагностувати паніку буде складніше: якщо драйвер DRM не запущений, ми опинимося на консолі VGA, яка може вмістити лише 24 рядки, і відповідне повідомлення ядра часто зникає; якщо драйвер DRM запущений, дисплей часто повністю псується під час паніки. Станом на Linux-6.12, жоден із спеціальних драйверів для основних моделей GPU насправді не підтримує це, але це підтримується «Простим драйвером фреймбуфера», який працює на фреймбуфері VESA (або EFI) до завантаження спеціального драйвера GPU. Якщо спеціальний драйвер GPU складено як модуль (а не як частину образу ядра) і не використовується `initramfs`, ця функція буде працювати нормально до моменту монтування кореневої файлової системи, і цього вже достатньо для надання інформації про більшість помилок конфігурації LFS, що викликають паніку (наприклад, неправильне налаштування `root=` у розділі 10.4, «Використання GRUB для налаштування процесу завантаження»).

Panic screen formatter

Встановіть цей `kmsg`, щоб переконатися, що останні рядки повідомлень ядра відображаються, коли відбувається паніка ядра. За замовчуванням, `user`, ядро показуватиме лише «зручне для користувача» повідомлення про паніку, яке не є корисним для діагностики. Третій варіант, `qr_code`, змусить ядро стиснути останні рядки повідомлень ядра в QR-код і відобразити його. QR-код може містити більше рядків повідомлень, ніж звичайний текст, і його можна декодувати за допомогою зовнішнього пристрою (наприклад, смартфона). Але для цього потрібен компілятор Rust, якого LFS не надає.

Mark VGA/VBE/EFI FB as generic system framebuffer та Simple framebuffer driver

Це дозволяє використовувати фреймбуфер VESA (або фреймбуфер EFI, якщо система LFS завантажується через UEFI) як пристрій DRM. Фреймбуфер VESA буде налаштований GRUB (або фреймбуфер EFI буде налаштований прошивкою UEFI), тому обробник паніки DRM може функціонувати до завантаження драйвера DRM, специфічного для GPU.

Enable legacy fbdev support for your modesetting driver та Framebuffer Console support

Вони потрібні для відображення консолі Linux на GPU, що керується драйвером DRI (Direct Rendering Infrastructure). Оскільки `CONFIG_DRM` (Direct Rendering Manager) увімкнено, ми також повинні увімкнути ці дві опції, інакше ми побачимо порожній екран після завантаження драйвера DRI.

Support x2apic

Підтримка роботи контролера переривань 64-бітних процесорів x86 у режимі x2APIC. x2APIC може бути увімкнено прошивкою на 64-бітних системах x86, і ядро без увімкненої цієї опції буде панікувати під час завантаження, якщо x2APIC увімкнено прошивкою. Ця опція не має впливу, але також не завдає шкоди, якщо x2APIC вимкнено прошивкою.

В деяких ситуаціях більш доцільним може бути використання команди **make oldconfig**. Докладнішу інформацію дивіться у файлі README.

За бажанням, можна пропустити конфігурацію ядра, скопіювавши файл конфігурації ядра `.config` з хост-системи (якщо він доступний) до розпакованого каталогу `linux-6.13.4`. Однак ми не рекомендуємо цей варіант. Часто краще переглянути всі меню конфігурації та створити конфігурацію ядра з нуля.

Скомпілюйте образ ядра та модулі:

make

Якщо використовуються модулі ядра, може знадобитися конфігурація модулів у `/etc/modprobe.d`. Інформація щодо модулів та конфігурації ядра міститься у розділі 9.3, «Огляд роботи з пристроями та модулями», а також у документації ядра у каталозі `linux-6.13.4/Documentation`. Також може бути цікавим `modprobe.d(5)`.

Якщо підтримка модулів не була вимкнена в конфігурації ядра, встановіть модулі за допомогою:

make modules_install

Після завершення компіляції ядра необхідно виконати додаткові кроки для завершення встановлення. Деякі файли потрібно скопіювати до каталогу `/boot`.

**Caution**

Якщо ви вирішили використовувати окремий розділ `/boot` для системи LFS (можливо, спільно з розділом `/boot` дистрибутива хоста), то скопійовані нижче файли слід розмістити саме там. Найпростіший спосіб зробити це — спочатку створити запис для `/boot` у файлі `/etc/fstab` (детальніше див. попередній розділ), а потім виконати наступну команду як користувач `root` у середовищі `chroot`:

mount /boot

Шлях до вузла пристрою опущений у команді, оскільки **mount** може прочитати його з `/etc/fstab`.

Шлях до образу ядра може відрізнятися залежно від використовуваної платформи. Назва файлу, наведена нижче, може бути змінена на вашу вподобу, але основа імені файла повинна бути `vmlinuz`, щоб бути сумісною з автоматичною настройкою процесу завантаження, описаною в наступному розділі. Наступна команда передбачає архітектуру x86:

cp -iv arch/x86/boot/bzImage /boot/vmlinuz-6.13.4-lfs-12.3

`System.map` — це файл символів для ядра. Він відображає точки входу кожної функції в API ядра, а також адреси структур даних ядра для працюючого ядра. Він використовується як ресурс під час розслідування проблем ядра. Виконайте наступну команду, щоб встановити файл карти:

cp -iv System.map /boot/System.map-6.13.4

Файл конфігурації ядра `.config`, створений на кроці **make menuconfig** вище, містить усі параметри конфігурації для ядра, яке щойно було скомпільовано. Рекомендується зберегти цей файл для подальшого використання:

```
cp -iv .config /boot/config-6.13.4
```

Встановіть документацію для ядра Linux:

```
cp -r Documentation -T /usr/share/doc/linux-6.13.4
```

Важливо зазначити, що файли в каталозі джерел ядра не належать користувачеві `root`. Коли пакет розпаковується як користувач `root` (як ми це зробили в `chroot`), файли мають ідентифікатори користувача та групи, які вони мали на комп'ютері пакувальника. Зазвичай це не є проблемою для встановлення інших пакетів, оскільки дерево джерел видаляється після встановлення. Однак дерево джерел Linux часто зберігається протягом тривалого часу. Через це існує ймовірність, що ідентифікатор користувача, який використовував пакувальник, буде призначений комусь на машині. Ця особа тоді матиме доступ для запису до джерел ядра.



Note

У багатьох випадках конфігурацію ядра потрібно буде оновити для пакетів, які будуть встановлені пізніше в BLFS. На відміну від інших пакетів, не потрібно видаляти дерево джерел ядра після встановлення новоствореного ядра.

Якщо дерево джерел ядра буде збережено, запустіть `chown -R 0:0` у каталозі `linux-6.13.4`, щоб переконатися, що всі файли належать користувачеві `root`.

Якщо ви оновлюєте конфігурацію та перекомпілюєте ядро з збереженого дерева джерел ядра, зазвичай **не слід** виконувати команду `make mrproper`. Ця команда видалить файл `.config` та всі файли `.o` з попередньої збірки. Незважаючи на те, що `.config` легко відновити з копії в `/boot`, видалення всіх файлів `.o` все одно є марною тратою часу: для простій зміни конфігурації часто потрібно (пере)збудувати лише кілька файлів `.o`, а система збірки ядра правильно пропустить інші файли `.o`, якщо вони не будуть видалені.

З іншого боку, якщо ви оновили GCC, вам слід виконати `make clean`, щоб видалити всі файли `.o` з попередньої збірки, інакше нова збірка може завершитися невдачею.



Warning

Деякі документи ядра рекомендують створити символічне посилання з `/usr/src/linux`, яке вказує на каталог джерел ядра. Це стосується ядер до серії 2.6 і **не повинно** створюватися в системі LFS, оскільки це може спричинити проблеми для пакетів, які ви, можливо, захочете зібрати після завершення базової системи LFS.

10.3.2. Налаштування порядку завантаження модулів Linux

У більшості випадків модулі Linux завантажуються автоматично, але іноді потрібні певні вказівки. Програма, яка завантажує модулі, `modprobe` або `insmod`, використовує для цього файл `/etc/modprobe.d/usb.conf`. Цей файл потрібно створити, щоб драйвери USB (`ehci_hcd`, `ohci_hcd` та

uhci_hcd), якщо вони були скомпільовані як модулі, завантажувалися в правильному порядку; ehci_hcd потрібно завантажувати перед ohci_hcd та uhci_hcd, щоб уникнути виведення попередження під час завантаження.

Створіть новий файл /etc/modprobe.d/usb.conf, виконавши наступну команду:

```
install -v -m755 -d /etc/modprobe.d
cat > /etc/modprobe.d/usb.conf << "EOF"
# Begin /etc/modprobe.d/usb.conf

install ohci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i ohci_hcd ; true
install uhci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i uhci_hcd ; true

# End /etc/modprobe.d/usb.conf
EOF
```

Встановіть документацію для ядра Linux:

дготуйтеся до компіляцію, виконавши наступну команду:

```
make mrproper
```

```
make install
```

10.3.3. Вміст Linux

Встановлені файли: config-6.13.4, vmlinuz-6.13.4-lfs-12.3, System.map-6.13.4

Встановлені директорії: /lib/modules, /usr/share/doc/linux-6.13.4

Короткі описи

config-6.13.4

Містить усі параметри конфігурації ядра

vmlinuz-6.13.4-lfs-12.3

Двигун системи Linux. При ввімкненні комп'ютера ядро є першою частиною операційної системи, яка завантажується. Воно виявляє та ініціалізує всі компоненти апаратного забезпечення комп'ютера, а потім робить ці компоненти доступними у вигляді дерева файлів для програмного забезпечення та перетворює один процесор на багатозадачну машину, здатну виконувати десятки програм одночасно

System.map-6.13.4

Список адрес і символів; він відображає точки входу та адреси всіх функцій і структур даних в ядрі

10.4. Використання GRUB для налаштування процесу завантаження



Note

Якщо ваша система підтримує UEFI і ви бажаєте завантажувати LFS з UEFI, вам слід пропустити інструкції на цій сторінці, але все одно вивчити синтаксис grub.cfg і метод визначення розділу у файлі з цієї сторінки, а також налаштувати GRUB з підтримкою UEFI, використовуючи інструкції, наведені на сторінці BLFS.

10.4.1. Вступ



Warning

Неправильна конфігурація GRUB може привести до того, що ваша система стане непрацездатною без альтернативного завантажувального пристрою, такого як CD-ROM або завантажувальний USB-накопичувач. Цей розділ не є обов'язковим для завантаження вашої системи LFS. Можливо, ви просто хочете змінити ваш поточний завантажувач, наприклад Grub-Legacy, GRUB2 або LILO.

Переконайтесь, що аварійний завантажувальний диск готовий до «рятування» комп'ютера, якщо він стане непридатним для використання (незавантажуваним). Якщо у вас ще немає завантажувального пристрою, ви можете його створити. Щоб процедура, описана нижче, працювала, вам потрібно перейти до BLFS і встановити xorriso з пакета libisoburn.

```
cd /tmp
grub-mkrescue --output=grub-img.iso
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

10.4.2. Конвенції іменування GRUB

GRUB використовує власну структуру іменування для дисків і розділів у формі (hdn,m) , де n — номер жорсткого диска, а m — номер розділу. Номери жорстких дисків починаються з нуля, але номери розділів починаються з одиниці для звичайних розділів (з п'яти для розширених розділів). Зверніть увагу, що це відрізняється від попередніх версій, де обидва номери починалися з нуля. Наприклад, розділ `sda1` для GRUB є $(hd0,1)$, а `sdb3` є $(hd1,3)$. На відміну від Linux, GRUB не вважає CD-ROM-диски жорсткими дисками. Наприклад, якщо ви використовуєте CD на `hd0` і другий жорсткий диск на `hd1`, цей другий жорсткий диск все одно буде $(hd1)$.

10.4.3. Налаштування конфігурації

GRUB працює, записуючи дані на перший фізичний трек жорсткого диска. Ця область не є частиною жодної файлової системи. Програми, що знаходяться там, отримують доступ до модулів GRUB у розділі завантаження. За замовчуванням це `/boot/grub/`.

Розташування розділу завантаження вибирає користувач, що впливає на конфігурацію. Однією з рекомендацій є створення окремого невеликого (рекомендований розмір — 200 МБ) розділу тільки для інформації про завантаження. Таким чином, кожна збірка, будь то LFS або якась комерційна дистрибуція, може отримати доступ до тих самих файлів завантаження, і доступ можна отримати з будь-якої завантаженої системи. Якщо ви вирішите це зробити, вам потрібно буде змонтувати окремий розділ, перемістити всі файли з поточного каталогу `/boot` (наприклад, ядро Linux, яке ви щойно зібрали в попередньому розділі) на новий розділ. Потім вам потрібно буде змонтувати розділ і знову змонтувати його як `/boot`. Якщо ви це зробите, не забудьте оновити `/etc/fstab`.

Залишення `/boot` на поточному розділі LFS також буде працювати, але конфігурація для декількох систем буде складнішою.

Використовуючи вищевказану інформацію, визначте відповідний позначник для кореневого розділу (або розділу завантаження, якщо використовується окремий розділ). У наступному прикладі передбачається, що кореневий (або окремий розділ завантаження) розділ — `sda2`.

Встановіть файли GRUB у `/boot/grub` і налаштуйте завантажувальний трек:

Warning

Наступна команда замінить поточний завантажувач. Не виконуйте цю команду, якщо це небажано, наприклад, якщо ви використовуєте сторонній менеджер завантаження для управління головним завантажувальним записом (MBR).

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

Note

Якщо система була завантажена за допомогою UEFI, `grub-install` спробує встановити файли для цілі `x86_64-efi`, але ці файли не були встановлені в розділі 8. У цьому випадку додайте `--target i386-pc` до команди вище.

10.4.4. Створення конфігураційного файлу GRUB

Створити файл `/boot/grub/grub.cfg`:

```
cat > /boot/grub/grub.cfg << "EOF"
# Begin /boot/grub/grub.cfg
set default=0
set timeout=5

insmod part_gpt
insmod ext2
set root=(hd0,2)
set gfxpayload=1024x768x32

menuentry "GNU/Linux, Linux 6.13.4-lfs-12.3" {
    linux /boot/vmlinuz-6.13.4-lfs-12.3 root=/dev/sda2 ro
}
EOF
```

Команди **insmod** завантажують модулі GRUB з іменами `part_gpt` та `ext2`. Незважаючи на назву, `ext2` насправді підтримує файлові системи `ext2`, `ext3` та `ext4`. Команда **grub-install** вбудувала деякі модулі в основний образ GRUB (встановлений в MBR або розділ GRUB BIOS) для доступу до інших модулів (в `/boot/grub/i386-pc`) без проблеми «яйце чи курка», тому в типовій конфігурації ці два модулі вже вбудовані і ці дві команди `insmod` нічого не зроблять. Але вони все одно не завдають шкоди і можуть знадобитися в деяких рідкісних конфігураціях.

Команда **set gfxpayload=1024x768x32** встановлює роздільну здатність і глибину кольору буфера кадрів VESA, що передається ядру. Це необхідно для драйвера SimpleDRM ядра, щоб використовувати буфер кадрів VESA. Ви можете використовувати іншу роздільну здатність або глибину кольору, які краще підходять для вашого монітора.

Note

З точки зору GRUB, файли ядра відносяться до використованого розділу. Якщо ви використовували окремий розділ `/boot`, видаліть `/boot` з наведеного вище рядка `linux`. Вам також потрібно буде змінити рядок `set root`, щоб він вказував на розділ завантаження.

Note

Позначення GRUB для розділу може змінитися, якщо ви додали або видали деякі диски (включаючи знімні диски, такі як USB-накопичувачі). Ця зміна може спричинити збій завантаження, оскільки `grub.cfg` посилається на деякі «старі» позначення. Якщо ви хочете уникнути такої проблеми, ви можете використовувати UUID розділу та UUID файлової системи замість позначення GRUB для визначення пристрою. Виконайте `lsblk -o UUID,PARTUUID,PATH,MOUNTPOINT`, щоб відобразити UUID ваших файлових систем (у стовпці UUID) та розділів (у стовпці PARTUUID). Потім замініть `set root=(hdx,y)` на `search --set=root --fs-uuid <UUID of the filesystem where the kernel is installed>`, а `root=/dev/sda2` замініть на `root=PARTUUID=<UUID of the partition where LFS is built>`.

Зверніть увагу, що UUID розділу повністю відрізняється від UUID файлової системи в цьому розділі. Деякі онлайн-ресурси можуть рекомендувати використовувати `root=UUID=<UUID filesystem>` замість `root=PARTUUID=<UUID partition>`, але для цього знадобиться `initramfs`, що виходить за межі LFS.

Ім'я вузла пристрою для розділу в `/dev` також може змінитися (це менш імовірно, ніж зміна позначення GRUB). Ви також можете замінити шляхи до вузлів пристроїв, таких як `/dev/sda1`, на `PARTUUID=<partition UUID>` в `/etc/fstab`, щоб уникнути потенційної помилки завантаження в разі зміни імені вузла пристрою.

GRUB — надзвичайно потужна програма, яка надає величезну кількість опцій для завантаження з широкого діапазону пристроїв, операційних систем та типів розділів. Існує також багато опцій для налаштування, таких як графічні заставки, відтворення звуків, введення за допомогою миші тощо. Детальна інформація про ці опції виходить за межі цього вступу.



Caution

Існує команда `grub-mkconfig`, яка може автоматично записати файл конфігурації. Вона використовує набір скриптів у `/etc/grub.d/` і знищить усі налаштування, які ви зробили. Ці скрипти призначені в першу чергу для дистрибутивів, що не є вихідними, і не рекомендуються для LFS. Якщо ви встановлюєте комерційний дистрибутив Linux, існує велика ймовірність, що ця програма буде запущена. Обов'язково зробіть резервну копію `grub.cfg`.

Глава 11. Кінець

11.1. Кінець

Молодець! Нова система LFS встановлена! Бажаємо вам успіхів у роботі з вашою новою, спеціально створеною для вас системою Linux.

Можливо, варто створити файл `/etc/lfs-release`. За допомогою цього файла вам (а також нам, якщо вам колись знадобиться допомога) буде дуже легко дізнатися, яка версія LFS встановлена в системі. Створіть цей файл, виконавши команду

```
echo 12.3 > /etc/lfs-release
```

Два файли, що описують встановлену систему, можуть використовуватися пакетами, які можуть бути встановлені в системі пізніше, або в бінарній формі, або шляхом їх компіляції.

Перший файл показує стан вашої нової системи стосовно Linux Standards Base (LSB). Щоб створити цей файл, виконайте:

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="12.3"
DISTRIB_CODENAME=""
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

Другий містить приблизно ту саму інформацію і використовується systemd та деякими графічними середовищами робочого столу. Щоб створити цей файл, виконайте:

```
cat > /etc/os-release << "EOF"
NAME="Linux From Scratch"
VERSION="12.3"
ID=lfs
PRETTY_NAME="Linux From Scratch 12.3"
VERSION_CODENAME=""
HOME_URL="https://www.linuxfromscratch.org/lfs/"
RELEASE_TYPE="stable"
EOF
```

Обов'язково налаштуйте поля «`DISTRIB_CODENAME`» та «`VERSION_CODENAME`», щоб система стала унікальною для вас.

11.2. Будьте враховані

Тепер, коли ви закінчили читати книгу, чи хочете ви стати користувачем LFS? Перейдіть на сторінку <https://www.linuxfromscratch.org/cgi-bin/lfscounter.php> і зареєструйтесь як користувач LFS, ввівши своє ім'я та першу версію LFS, яку ви використовували.

Давайте перезавантажимося в LFS.

11.3. Перезавантаження системи

Тепер, коли все програмне забезпечення встановлено, настав час перезавантажити комп'ютер. Однак, є ще кілька речей, які необхідно перевірити. Ось кілька пропозицій:

- Встановіть необхідне програмне забезпечення, якщо драйвер ядра для вашого обладнання вимагає певних файлів програмного забезпечення для належного функціонування.
- Переконайтесь, що для користувача `root` встановлено пароль.
- На цьому етапі також доцільно переглянути наступні файли конфігурації.
 - `/etc/bashrc`
 - `/etc/dircolors`
 - `/etc/fstab`
 - `/etc/hosts`
 - `/etc/inputrc`
 - `/etc/profile`
 - `/etc/resolv.conf`
 - `/etc/vimrc`
 - `/root/.bash_profile`
 - `/root/.bashrc`
 - `/etc/sysconfig/ifconfig.eth0`

Тепер, коли ми це з'ясували, перейдемо до першого запуску нашої нової інсталяції LFS! Спочатку вийдіть із середовища `chroot`:

```
logout
```

Потім від'єднайте віртуальні файлові системи:

```
umount -v $LFS/dev/pts
mountpoint -q $LFS/dev/shm && umount -v $LFS/dev/shm
umount -v $LFS/dev
umount -v $LFS/run
umount -v $LFS/proc
umount -v $LFS/sys
```

Якщо було створено кілька розділів, перед тим, як від'єднати основний розділ, від'єднайте інші розділи, як показано нижче:

```
umount -v $LFS/home
umount -v $LFS
```

Від'єднайте саму файлову систему LFS:

```
umount -v $LFS
```

Тепер перезавантажте систему.

Якщо завантажувач GRUB був налаштований, як описано раніше, меню буде налаштоване на автоматичне завантаження LFS 12.3.

Після завершення перезавантаження система LFS буде готова до використання. Ви побачите простий запит «`login:`». На цьому етапі ви можете перейти до книги BLFS, де ви зможете додати більше програмного забезпечення відповідно до своїх потреб.

Якщо перезавантаження **не вдалося**, час усунути неполадки. Поради щодо вирішення початкових проблем із завантаженням див. <https://www.linuxfromscratch.org/lfs/troubleshooting.html>.

11.4. Додаткові ресурси

Дякуємо за прочитання цієї книги про LFS. Сподіваємося, що вона була для вас корисною і ви дізналися більше про процес створення системи.

Тепер, коли система LFS встановлена, ви, можливо, задаєтесь питанням: «Що далі?» Щоб відповісти на це питання, ми склали для вас список ресурсів.

- Технічне обслуговування

Про помилки та проблеми безпеки регулярно повідомляється для всього програмного забезпечення. Оскільки система LFS компілюється з вихідного коду, ви повинні самостійно стежити за такими повідомленнями. Існує кілька онлайн-ресурсів, які відстежують такі повідомлення, деякі з яких наведено нижче:

- *LFS Security Advisories*

Це список вразливостей безпеки, виявлених у книзі LFS після її публікації.

- *Open Source Security Mailing List*

Це список розсилки для обговорення вразливостей безпеки, концепцій та практик у спільноті відкритого програмного забезпечення.

- Підказки LFS

Підказки LFS — це збірка навчальних документів, надісланих волонтерами спільноти LFS. Підказки доступні за адресою <https://www.linuxfromscratch.org/hints/downloads/files/>.

- Списки розсилки

Існує кілька списків розсилки LFS, на які ви можете підписатися, якщо вам потрібна допомога, ви хочете бути в курсі останніх подій, хочете взяти участь у проекті тощо. Дивіться розділ 1 — Списки розсилки для отримання додаткової інформації.

- The Linux Documentation Project

Метою Проекту документації Linux (TLDL) є співпраця з усіх питань, пов'язаних з документацією Linux. TLDL містить велику колекцію HOWTO, посібників та сторінок man. Він розташований за адресою <https://tldp.org/>.

11.5. Початок роботи після LFS

11.5.1. Вирішення, що робити далі

Тепер, коли LFS завершено і ви маєте завантажувальну систему, що робити далі? Наступним кроком є рішення про те, як її використовувати. Загалом, є дві широкі категорії, які слід врахувати: робоча станція або

сервер. Звичайно, ці категорії не є взаємовиключними. Програми, необхідні для кожної категорії, можна об'єднати в одну систему, але давайте розглянемо їх поокремо.

Сервер є більш простою категорією. Зазвичай він складається з веб-сервера, такого як Apache HTTP Server, та сервера бази даних, такого як MariaDB. Однак можливі й інші сервіси. Операційна система, вбудована в одноразовий пристрій, відноситься до цієї категорії.

З іншого боку, робоча станція є набагато складнішою. Зазвичай вона вимагає графічного середовища користувача, такого як LXDE, XFCE, KDE або Gnome, що базується на базовому графічному середовищі та декількох графічних додатках, таких як веб-браузер Firefox, поштовий клієнт Thunderbird або офісний пакет LibreOffice. Ці додатки вимагають набагато (кілька сотень, залежно від бажаних можливостей) більше пакетів допоміжних додатків та бібліотек.

Крім вищезазначеного, існує набір програм для управління всіма видами систем. Ці програми всі наведені в книзі BLFS. Не всі пакети потрібні в кожному середовищі. Наприклад, *dhcpcd*, як правило, не підходить для сервера, а *wireless_tools*, як правило, корисний тільки для ноутбуків.

11.5.2. Робота в базовому середовищі LFS

Коли ви вперше завантажуєте LFS, у вас є всі внутрішні інструменти для створення додаткових пакетів. На жаль, користувацьке середовище є досить обмеженим. Є кілька способів поліпшити це:

11.5.2.1. Робота з хостом LFS в chroot

Цей метод забезпечує повне графічне середовище, в якому доступні повнофункціональний браузер та можливості копіювання/вставлення. Цей метод дозволяє використовувати такі програми, як версія *wget* хоста, для завантаження джерел пакетів у місце, доступне під час роботи в середовищі *chroot*.

Для того, щоб правильно створювати пакети в *chroot*, вам також потрібно пам'ятати про необхідність монтувати віртуальні файлові системи, якщо вони ще не змонтовані. Один із способів зробити це — створити скрипт на системі **HOST**:

```
cat > ~/mount-virt.sh << "EOF"
#!/bin/bash

function mountbind
{
    if ! mountpoint $LFS/$1 >/dev/null; then
        $SUDO mount --bind /$1 $LFS/$1
        echo $LFS/$1 mounted
    else
        echo $LFS/$1 already mounted
    fi
}

function mounttype
{
    if ! mountpoint $LFS/$1 >/dev/null; then
        $SUDO mount -t $2 $3 $4 $5 $LFS/$1
        echo $LFS/$1 mounted
    else
        echo $LFS/$1 already mounted
    fi
}
```

```

echo $LFS/$1 already mounted
fi
}

if [ $EUID -ne 0 ]; then
  SUDO=sudo
else
  SUDO=""
fi

if [ x$LFS == x ]; then
  echo "LFS not set"
  exit 1
fi

mountbind dev
mounttype dev/pts      devpts      devpts -o gid=5,mode=620
mounttype proc          proc        proc
mounttype sys           sysfs      sysfs
mounttype run            tmpfs      run
if [ -h $LFS/dev/shm ]; then
  install -v -d -m 1777 $LFS$(realpath /dev/shm)
else
  mounttype dev/shm tmpfs tmpfs -o nosuid,nodev
fi

#mountbind usr/src
#mountbind boot
#mountbind home
EOF

```

Зверніть увагу, що останні три команди в скрипті закомментовані. Вони корисні, якщо ці каталоги змонтовані як окремі розділи на хост-системі і будуть змонтовані під час завантаження готової системи LFS/BLFS.

Скрипт можна запустити за допомогою команди **bash ~/mount-virt.sh** як звичайний користувач (рекомендовано) або як `root`. Якщо запускати як звичайний користувач, на хост-системі потрібно встановити `sudo`.

Ще одне питання, на яке звертає увагу скрипт, — це місце зберігання завантажених файлів пакетів. Це місце є довільним. Воно може знаходитися у звичайному домашньому каталозі користувача, наприклад `/sources`, або в глобальному місці, наприклад `/usr/src`. Ми рекомендуємо не змішувати джерела BLFS і LFS у (з середовища `chroot`) `/sources`. У будь-якому випадку пакети повинні бути доступними всередині середовища `chroot`.

Остання зручна функція, представлена тут, полягає в оптимізації процесу входу в середовище `chroot`. Це можна зробити за допомогою псевдоніма, розміщеного у файлі `~/.bashrc` користувача на хост-системі:

```
alias lfs='sudo /usr/sbin/chroot /mnt/lfs /usr/bin/env -i HOME=/root TERM="$TERM" PS1="\u:\w\\\$ "
PATH=/usr/bin:/usr/sbin /bin/bash --login'
```

Цей псевдонім є дещо складним через використання лапок та рівнів символів зворотного слешу. Він повинен бути вказаний в одному рядку. Для зручності демонстрації вищевказанна команда була розділена на дві частини.

11.5.2.2. Працюйте віддалено через ssh

Цей метод також забезпечує повне графічне середовище, але спочатку вимагає встановлення sshd на системі LFS, зазвичай у chroot. Він також вимагає другого комп'ютера. Цей метод має перевагу у вигляді простоти, оскільки не вимагає складності середовища chroot. Він також використовує ядро, побудоване LFS, для всіх додаткових пакетів і все одно забезпечує повну систему для встановлення пакетів.

Ви можете використовувати команду **scp** для завантаження джерел пакетів, які будуть скомпільовані в системі LFS. Якщо ви хочете завантажити джерела безпосередньо в систему LFS, встановіть *libtasn1*, *p11-kit*, *make-ca* та *wget* в chroot (або завантажте їхні джерела за допомогою **scp** після завантаження системи LFS).

11.5.2.3. Працюйте віддалено через ssh

Цей метод вимагає встановлення *libtasn1*, *p11-kit*, *make-ca*, *wget*, *gpm* та *links* (або *lynx*) у chroot, а потім перезавантаження у нову систему LFS. На цьому етапі система за замовчуванням має шість віртуальних консолей. Перемикання консолей здійснюється за допомогою комбінації клавіш **Alt+Fx**, де **Fx** — це клавіша від **F1** до **F6**. Комбінації **Alt+←** та **Alt+→** також змінюють консоль.

На цьому етапі ви можете увійти в дві різні віртуальні консолі та запустити браузер *links* або *lynx* в одній консолі, а *bash* в іншій. GPM дозволяє копіювати команди з браузера за допомогою лівої кнопки миші, перемикатися між консолями та вставляти їх в іншу консоль.



Note

До речі, перемикання віртуальних консолей також можна здійснювати з екземпляра X Window за допомогою комбінації клавіш **Ctrl+Alt+Fx**, але операція копіювання мишею не працює між графічним інтерфейсом і віртуальною консольлю. Ви можете повернутися до дисплея X Window за допомогою комбінації **Ctrl+Alt+Fx**, де **Fx** зазвичай є **F1**, але може бути і **F7**.