Linux From Scratch

Versjon 11.1-systemd

Publisert 1. Mars 2022

Laget av Gerard Beekmans
Administrerende redaktør: Bruce Dubbs
Redaktør: Douglas R. Reno
Redaktør: DJ Lucas

Linux From Scratch: Versjon 11.1-systemd: Publisert 1. Mars 2022

by Laget av Gerard Beekmans, Administrerende redaktør: Bruce Dubbs, Redaktør: Douglas R. Reno, and Redaktør: DJ Lucas

Table of Contents

Forord viii
i. Forord viii
ii. Publikum viii
iii. LFS målarkitekturer ix
iv. Forutsetninger x
v. LFS og standarder x
vi. Begrunnelse for pakker i boken xi
vii. Typografi xvii
viii. Struktur xviii
ix. Errata og sikkerhetsråd xix
I. Introduksjon 1
1. Introduksjon 2
1.1. Hvordan bygge et LFS-system 2
1.2. Hva er nytt siden forrige utgivelse 2
1.3. Endringslogg 4
1.4. Ressurser 8
1.5. Hjelp 8
II. Forbereder for byggingen 11
2. Klargjøring av vertssystemet 12
2.1. Introduksjon 12
2.2. Systemkrav for vert 12
2.3. Bygge LFS i etapper 14
2.4. Opprette en ny partisjon 15
2.5. Opprette et filsystem på partisjonen 17
2.6. Stille inn $LFS variabelen 17
2.7. Montering av den nye partisjonen 18
3. Pakker og oppdateringer 20
3.1. Introduksjon 20
3.2. Alle pakker 21
3.3. Nødvendige oppdateringer 29
4. Siste forberedelser 31
4.1. Introduksjon 31
4.2. Opprette et begrenset mappeoppsett i LFS filsystemet 31
4.3. Legge til LFS brukeren 31
4.4. Sette opp miljøet 32
4.5. Om SBU 34
4.6. Om testpakkene 35
III. Bygge LFS Kryssverktøykjede og midlertidige verktøy 36
Viktig foreløpig materiale xxxvii
i. Introduksjon xxxvii
ii. Verktøykjedens tekniske merknader xxxvii
iii. Generelle kompileringsinstruksjoner xli
5. Kompilere en kryssverktøykjede 43
5.1. Introduksjon 43
5.2. Binutils-2.38 - Pass 1 44

5.3. GCC-11.2.0 - Pass 1 46
5.4. Linux-5.16.9 API Headers 49
5.5. Glibc-2.35 50
5.6. Libstdc++ from GCC-11.2.0, Pass 1 53
6. Krysskompilering av midlertidige verktøy 55
6.1. Introduksjon 55
6.2. M4-1.4.19 56
6.3. Ncurses-6.3 57
6.4. Bash-5.1.16 59
6.5. Coreutils-9.0 60
6.6. Diffutils-3.8 61
6.7. File-5.41 62
6.8. Findutils-4.9.0 63
6.9. Gawk-5.1.1 64
6.10. Grep-3.7 65
6.11. Gzip-1.11 66
6.12. Make-4.3 67
6.13. Patch-2.7.6 68
6.14. Sed-4.8 69
6.15. Tar-1.34 70
6.16. Xz-5.2.5 71
6.17. Binutils-2.38 - Pass 2 72
6.18. GCC-11.2.0 - Pass 2 73
7. Gå inn i Chroot og bygge ytterligere midlertidige verktøy 75
7.1. Introduksjon 75
7.2. Skifte eierskap 75
7.3. Klargjøring av virtuelle kjernefilsystemer 75
7.4. Gå inn i Chroot miljøet 76
7.5. Opprette mapper 77
7.6. Opprette essensielle filer og symbolkoblinger 78
7.7. Libstdc++ from GCC-11.2.0, Pass 2 82
7.8. Gettext-0.21 84
7.9. Bison-3.8.2 85
7.10. Perl-5.34.0 86
7.11. Python-3.10.2 87
7.12. Texinfo-6.8 88
7.13. Util-linux-2.37.4 89
7.14. Opprydding og lagring av det midlertidige systemet 91
IV. Bygge LFS systemet 93
8. Installere grunnleggende systemprogramvare 94
8.1. Introduksjon 94
8.2. Pakkehåndtering 95
8.3. Man-pages-5.13 99
8.4. Iana-Etc-20220207 100
8.5. Glibc-2.35 101
8.6. Zlib-1.2.11 109
8.7. Bzip2-1.0.8 110

8.8. Xz-5.2.5 112
8.9. Zstd-1.5.2 114
8.10. File-5.41 115
8.11. Readline-8.1.2 116
8.12. M4-1.4.19 118
8.13. Bc-5.2.2 119
8.14. Flex-2.6.4 120
8.15. Tcl-8.6.12 121
8.16. Expect-5.45.4 123
8.17. DejaGNU-1.6.3 124
8.18. Binutils-2.38 125
8.19. GMP-6.2.1 128
8.20. MPFR-4.1.0 130
8.21. MPC-1.2.1 131
8.22. Attr-2.5.1 132
8.23. Acl-2.3.1 133
8.24. Libcap-2.63 134
8.25. Shadow-4.11.1 135
8.26. GCC-11.2.0 139
8.27. Pkg-config-0.29.2 144
8.28. Ncurses-6.3 145
8.29. Sed-4.8 148
8.30. Psmisc-23.4 149
8.31. Gettext-0.21 150
8.32. Bison-3.8.2 152
8.33. Grep-3.7 153
8.34. Bash-5.1.16 154
8.35. Libtool-2.4.6 156
8.36. GDBM-1.23 157
8.37. Gperf-3.1 158
8.38. Expat-2.4.6 159
8.39. Inetutils-2.2 160
8.40. Less-590 162
8.41. Perl-5.34.0 163
8.42. XML::Parser-2.46 166
8.43. Intltool-0.51.0 167
8.44. Autoconf-2.71 168
8.45. Automake-1.16.5 170
8.46. OpenSSL-3.0.1 171
8.47. Kmod-29 173
8.48. Libelf fra Elfutils-0.186 175
8.49. Libffi-3.4.2 176
8.50. Python-3.10.2 177
8.51. Ninja-1.10.2 179
8.52. Meson-0.61.1 181
8.53. Coreutils-9.0 182
8.54. Check-0.15.2 187

8.55. Diffutils-3.8 188
8.56. Gawk-5.1.1 189
8.57. Findutils-4.9.0 190
8.58. Groff-1.22.4 192
8.59. GRUB-2.06 195
8.60. Gzip-1.11 197
8.61. IPRoute2-5.16.0 198
8.62. Kbd-2.4.0 200
8.63. Libpipeline-1.5.5 202
8.64. Make-4.3 203
8.65. Patch-2.7.6 204
8.66. Tar-1.34 205
8.67. Texinfo-6.8 206
8.68. Vim-8.2.4383 208
8.69. MarkupSafe-2.0.1 211
8.70. Jinja2-3.0.3 212
8.71. Systemd-250 213
8.72. D-Bus-1.12.20 218
8.73. Man-DB-2.10.1 220
8.74. Procps-ng-3.3.17 223
8.75. Util-linux-2.37.4 225
8.76. E2fsprogs-1.46.5 230
8.77. Om feilsøking av symboler 233
8.78. Stripping 233
8.79. Rydde opp 235
9. Systemkonfigurasjon 236
9.1. Introduksjon 236
9.2. Generell nettverkskonfigurasjon 236
9.3. Oversikt over enhets- og modulhåndtering 240
9.4. Administrere enheter 243
9.5. Konfigurering av systemklokken 244
9.6. Konfigurering av Linux konsollen 245
9.7. Konfigurere systemlokaliteten 246
9.8. Opprette /etc/inputrc filen 248
9.9. Opprette /etc/shells filen 250
9.10. Systemd bruk og konfigurasjon 250
10. Gjøre LFS systemet oppstartbart 254
10.1. Introduksjon 254
10.2. Opprette /etc/fstab filen 254
10.3. Linux-5.16.9 256
10.4. Bruke GRUB til å sette opp oppstartsprosessen 261
11. Slutten 263
11.1. Slutten 263
11.2. Bli regnet med 263
11.3. Omstart av systemet 264
11.4. Hva nå? 265
V. Vedlegg 267

A. Akronymer og begreper 268
B. Anerkjennelser 271
C. Avhengigheter 274
D. LFS lisenser 290
D.1. Creative Commons License 290
D.2. The MIT License 294
Index 295

Forord

Forord

Min reise for å lære og bedre forstå Linux begynte tilbake i 1998. Jeg hadde nettopp installert min første Linux-distribusjon og hadde raskt blitt fascinert av hele konseptet og filosofien bak Linux.

Det er alltid mange måter å utføre en enkelt oppgave på. Det samme kan sies om Linux-distribusjoner. Svært mange har eksistert opp gjennom årene. Noen eksisterer fortsatt, noen har forvandlet seg til noe annet, mens andre har blitt henvist til våre minner. De gjør alle ting annerledes for å passe behovene til deres målgruppe. Fordi det eksisterer så mange forskjellige måter å oppnå det samme sluttmålet , begynte jeg å innse at jeg ikke lenger måtte være begrenset av noen gjennomføring. Før vi oppdaget Linux, stilte vi rett og slett opp med problemer i andre operativsystemer siden du ikke hadde noe valg. Det var hva det var, enten du likte det eller ikke. Med Linux begynte konseptet med valg å dukke opp. Hvis du ikke likte noe, du var fri, til og med oppmuntret, til å endre det.

Jeg prøvde en rekke distribusjoner og kunne ikke bestemme meg for noen. De var flotte systemer i seg selv. Det var ikke et spørsmål om rett og feil lenger. Det var blitt et spørsmål om personlig smak. Med allle de valgene tilgjengelig, ble det klart at det ikke ville være en eneste system som ville være perfekt for meg. Så jeg satte meg for å lage min egen Linux system som fullt ut samsvarer med mine personlige preferanser.

For å virkelig gjøre det til mitt eget system, bestemte jeg meg for å kompilere alt fra kildekode i stedet for å bruke forhåndskompilerte binære pakker. Dette “perfekt” Linux-system vil ha styrken til forskjellige systemer uten deres opplevde svakheter. Først var tanken snarere skremmende. Jeg forble forpliktet til ideen om at et slikt system kunne bli bygget.

Etter å ha sortert gjennom problemer som sirkulære avhengigheter og kompileringsfeil , bygget jeg endelig et spesialbygd Linux-system. Det var fullt operativ og perfekt brukbart som alle andre Linux-systemer ute der på den tiden. Men det var min egen skapelse. Det var veldig tilfredsstillende å har satt sammen et slikt system selv. Det eneste bedre ville ha vært å lage hvert stykke programvare selv. Dette var det nest beste .

Da jeg delte mine mål og erfaringer med andre medlemmer av Linux samfunnet, ble det tydelig at det var en vedvarende interesse for disse ideer. Det ble raskt klart at slike spesialbygde Linux-systemer tjener ikke bare for å møte brukerspesifikke krav, men også tjene som en ideell læringsmulighet for programmerere og systemadministratorer forbedre deres (eksisterende) Linux-ferdigheter. Ut fra denne utvidede interessen *Linux From Scratch Project* ble født.

Denne Linux From Scratch boken er den sentrale kjernen rundt det prosjektet. Den gir bakgrunnen og instruksjonene som er nødvendige for deg å designe og bygge ditt eget system. Mens denne boken gir en mal som vil resultere i et korrekt fungerende system står du fritt til å endre instruksjonene til å passer deg selv, som delvis er en viktig del av dette prosjektet. Du forblir i kontroll; vi gir bare en hjelpende hånd for å komme i gang på din egen reise.

Jeg håper inderlig at du vil ha en flott tid med å jobbe med din egen Linux From Scratch system og nyte de mange fordelene ved å ha et system som er virkelig din egen.

--
Gerard Beekmans
gerard@linuxfromscratch.org

Publikum

Det er mange grunner til at du ønsker å lese denne boken. Et av spørsmålene mange spørrer seg er, “hvorfor gå gjennom alt bryet med å manuelt bygge et Linux system fra bunnen av når du bare kan laste ned og installere en eksisterende?”

En viktig grunn til dette prosjektets eksistens er å hjelpe deg med å lære hvordan et Linux system fungerer fra innsiden og ut. Å bygge et LFS system hjelper å demonstrere hva som får Linux til å virke, og hvordan ting fungerer sammen og avhenger av hverandre. Noe av det beste denne læringsopplevelsen kan gi er muligheten til å tilpasse et Linux system for å passe dine egne unike behov.

En annen viktig fordel med LFS er at den lar deg ha mer kontroll over systemet uten å stole på andres Linux implementering. Med LFS, du er i førersetet og dikterer alle aspekter av systemet.

LFS lar deg lage svært kompakte Linux systemer. Ved installasjon av vanlige distribusjoner, blir du ofte tvunget til å installere svært mange programmer som sannsynligvis aldri blir brukt eller forstått. Disse programmene sløser ressurser. Du kan hevde at det med dagens harddisk og CPUer, f.eks ressurser er ikke lenger en vurdering. Noen ganger er du imidlertid fortsatt begrenset av størrelseshensyn om ikke annet. Tenk på oppstartbar CDer, USB-pinner og innebygde systemer. Det er områder hvor LFS kan være gunstig.

En annen fordel med et spesialbygd Linux system er sikkerhet. Ved å kompilere hele systemet fra kildekoden, har du fullmakt til å revidere alt og bruk alle sikkerhetsoppdateringene du ønsker. Det er ikke lenger nødvendig å vente på at noen andre skal kompilere binære pakker som fikser et sikkerhetshull. Med mindre du undersøker oppdateringen og implementerer den selv, har du ingen garantier på at den nye binære pakken ble bygget riktig og løser pråblemet tilstrekkelig.

Målet med Linux From Scratch er å bygge en komplett og brukbart system på fundamentnivå. Hvis du ikke ønsker å bygge ditt eget Linux system fra bunnen av kan du likevel ha nytte av informasjonen i denne boken.

Det er for mange andre gode grunner til å bygge ditt eget LFS-system til liste dem alle her. Til syvende og sist er utdanning den desidert mest kraftfulle av grunner. Når du fortsetter i LFS-opplevelsen din, vil du oppdage kraften som informasjon og kunnskap virkelig gir.

LFS målarkitekturer

Det primære målarkitekturene til LFS er AMD/Intel x86 (32-bit) og x86_64 (64-bit) CPUer. På den annen side er instruksjonene i denne boken også kjent for å fungere, med noen modifikasjoner, med Power PC og ARM CPUer. Hovedforutsetningen for å bygge et system som bruker en av disse CPUene, i i tillegg til de på neste side, er et eksisterende Linux-system som f.eks tidligere LFS installasjon, Ubuntu, Red Hat/Fedora, SuSE eller annen distribusjon som retter seg mot arkitekturen du har. Vær også oppmerksom på at en 32-bit distribusjon kan installeres og brukes som et vertssystem på en 64-bit AMD/Intel datamaskin.

For å bygge LFS, gevinsten ved å bygge på et 64-bitssystem sammenlignet med et 32-bits system er minimal. For eksempel, i en testbygging av LFS-9.1 på et Core i7-4790 CPU-basert system, ved bruk av 4 kjerner ble følgende statistikk målt:

```
Arkitektur Byggetid     Byggestørrelse
32-bit       239.9 minutes  3.6 GB
64-bit       233.2 minutes  4.4 GB
```

Som du kan se, på den samme maskinvaren, er 64-bits bygg bare 3% raskere og er 22 % større enn 32-bits bygg. Hvis du planlegger å bruke LFS som en LAMP server, eller en brannmur, kan en 32-bits CPU stort sett være tilstrekkelig. På den andre siden, trenger flere pakker i BLFS nå mer enn 4 GB RAM for å bygges og/eller å kjøre, slik at hvis du planlegger å bruke LFS som skrivebord, så anbefaler LFS-forfatterne å bygge på et 64-bitssystem.

Standard 64-bits bygg som et resultat av LFS regnes som et “rent” 64-bits system. Det vil si at den bare støtter 64-biters kjørbare filer . Å bygge et “flerarkitekturs” system krever kompilering av mange applikasjoner to ganger, én gang for et 32-bitssystem og én gang for et 64-bitssystem. Dette støttes ikke direkte i LFS fordi det ville forstyrre pedagogisk mål

om å gi instruksjonene som trengs for et enkelt grunnleggende Linux-system. Noen LFS/BLFS-redaktører opprettholder en forgrening av LFS for flerarkitektur, som er tilgjengelig på *https://www.linuxfromscratch.org/~thomas/multilib/index.html*. Men det er et avansert tema.

Forutsetninger

Å bygge et LFS system er ikke en enkel oppgave. Det krever en viss nivå av eksisterende kunnskap om Unix systemadministrasjon for å løse problemer og utfør kommandoene som er oppført riktig. Spesielt som en absolutt minimum, bør du allerede ha muligheten til å bruke kommandolinje (skall) for å kopiere eller flytte filer og kataloger, liste katalog og filinnhold, og endre gjeldende katalog. Det forventes også at du har rimelig kunnskap om bruk og installasjon av Linux programvare.

Fordi LFS boka antar *i det minste* dette grunnleggende ferdighetsnivået, er det usannsynlig at de ulike LFS støtteforaene vil kunne gi deg mye hjelp på disse områdene. Du vil finne at dine spørsmål angående slik grunnleggende kunnskap vil sannsynligvis forbli ubesvart eller du vil ganske enkelt bli henvist til LFS essensielle forhåndsleseliste.

Før du bygger et LFS system, anbefaler vi å lese følgende:

- Programvare-bygging-HOWTO *http://www.tldp.org/HOWTO/Software-Building-HOWTO.html*

  Dette er en omfattende veiledning for bygging og installasjon av “generiske” Unix-programvarepakker under Linux. Selv om det ble skrevet for en tid siden, gir den fortsatt en god oppsummering av grunnleggende teknikker som trengs for å bygge og installere programvare.

- Nybegynnerveiledning for å installere fra kilden *http://moi.vonos.net/linux/beginners-installing-from-source/*

  Denne veiledningen gir en god oppsummering av grunnleggende ferdigheter og teknikker som trengs for å bygge programvare fra kildekode.

LFS og standarder

Strukturen til LFS følger Linux standarder så tett som mulig. De primære standardene er:

- *POSIX.1-2008.*
- *Standard for filsystemhierarki (FHS) Version 3.0*
- *Linux Standard base (LSB) Version 5.0 (2015)*

  LSB har fire separate standarder: Kjerne, Skrivebord, Kjøretidsspråk og bildebehandling. I tillegg til generiske krav er det også arkitekturspesifikke krav. Det er også to områder for prøvebruk: Gtk3 og grafikk. LFS forsøker å tilpasse seg arkitekturer omtalt i forrige avsnitt.

Note

Mange mennesker er ikke enige i kravene til LSB. Hovedformålet med å definere det er å sikre at proprietær programvare vil kunne installeres og kjøres riktig på et kompatibelt system. Siden LFS er kildebasert, har brukeren full kontroll over hvilke pakker som er ønsket og mange velger å ikke installere noen pakker som er spesifisert av LSB.

Å opprette et komplett LFS system som er i stand til å bestå LSB sertifiseringstester er mulig, men ikke uten mange tilleggspakker som er utenfor omfanget av LFS. Disse tilleggspakkene har installasjonsinstruksjoner i BLFS.

Pakker levert av LFS som tilfredsstiller LSB kravene

LSB Kjerne:	Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux, Zlib
LSB Skrivebord:	None
LSB Kjøretidsspråk:	Perl, Python
LSB Bildebehandling:	None
LSB Gtk3 og LSB Grafikk (Prøvebruk):	None

Pakker levert av BLFS som tilfredsstiller LSB kravene

LSB Kjerne:	At, Batch (a part of At), Cpio, Ed, Fcrontab, LSB-Tools, NSPR, NSS, PAM, Pax, Sendmail (or Postfix or Exim), time
LSB Skrivebord:	Alsa, ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Gdk-pixbuf, Glib2, GTK+2, Icon-naming-utils, Libjpeg-turbo, Libpng, Libtiff, Libxml2, MesaLib, Pango, Xdg-utils, Xorg
LSB Kjøretidsspråk:	Libxml2, Libxslt
LSB Bildebehandling:	CUPS, Cups-filters, Ghostscript, SANE
LSB Gtk3 and LSB Grafikk (Prøvebruk):	GTK+3

Pakker som ikke er levert av LFS eller BLFS nødvendige for å tilfredsstille LSB kravene

LSB Kjerne:	None
LSB Skrivebord:	Qt4 (but Qt5 is provided)
LSB Kjøretidsspråk:	None
LSB Bildebehandling:	None
LSB Gtk3 and LSB Grafikk (Prøvebruk):	None

Begrunnelse for pakker i boken

Som nevnt tidligere er målet med LFS å bygge en komplett og brukbar system på fundamentnivå. Dette inkluderer alle pakker som trengs for å replikere seg selv samtidig som det gir en relativt minimal base å tilpasse for et mer komplett system basert på brukerens valg. Dette betyr ikke at LFS er det minste systemet som er mulig å bygge. Flere viktige pakker er inkludert som ikke er strengt påkrevd. Listene nedenfor dokumenterer begrunnelsen for hver pakke i boken.

- Acl

  Denne pakken inneholder verktøy for å administrere tilgangskontrolllister, som brukes til å definere mer finkornet skjønnsmessige tilgangsrettigheter for filer og kataloger.

- Attr

  Denne pakken inneholder programmer for administrasjon av utvidede attributter på filsystemobjekter.

- Autoconf

Denne pakken inneholder programmer for å produsere skallskript som automatisk kan konfigurere kildekoden fra en utviklermal . Det er ofte nødvendig for å gjenoppbygge en pakke etter oppdateringer til byggeprosedyrene.

- Automake

  Denne pakken inneholder programmer for å generere Make filer fra en mal. Det er ofte nødvendig for å gjenoppbygge en pakke etter oppdateringer til byggeprosedyrene.

- Bash

  Denne pakken tilfredsstiller et LSB-kjernekrav for å gi et Bourne Shell grensesnitt til systemet. Det ble valgt over andre skallpakker på grunn av dens vanlige bruk og omfattende funksjoner utover grunnleggende skallfunksjoner.

- Bc

  Denne pakken gir et vilkårlig presisjons numerisk behandlingsspråk. Den tilfredsstiller et krav som er nødvendig når du bygger Linux kjernen.

- Binutils

  Denne pakken inneholder en linker, en assembler og annet verktøy for håndtering av objektfiler. Programmene i denne pakken er nødvendig for å kompilere de fleste pakkene i et LFS system og videre.

- Bison

  Denne pakken inneholder GNU-versjonen av yacc (Yet Another Compiler Compiler) nødvendig for å bygge flere andre LFS programmer.

- Bzip2

  Denne pakken inneholder programmer for komprimering og dekomprimering av filer. Det kreves for å dekomprimere mange LFS pakker.

- Check

  Denne pakken inneholder et testmiljø for andre programmer.

- Coreutils

  Denne pakken inneholder en rekke viktige programmer for visning og manipulering av filer og mapper. Disse programmene trengs for kommandolinjefilbehandling, og er nødvendige for installasjons prosedyrer for hver pakke i LFS.

- D-Bus

  Denne pakken inneholder programmer for å implementere et meldingsbusssystem, som er en enkel måte for programmer å snakke med hverandre på.

- DejaGNU

  Denne pakken inneholder et rammeverk for å teste andre programmer.

- Diffutils

  Denne pakken inneholder programmer som viser forskjellene mellom filer eller mapper. Disse programmene kan brukes til å lage oppdateringer (patcher), og brukes også i mange pakkers byggeprosedyrer.

- E2fsprogs

  Denne pakken inneholder verktøyene for å håndtere ext2, ext3 og ext4 filsystemer. Disse er de mest vanlige og grundig testede filsystemer som Linux støtter.

- Expat

  Denne pakken inneholder et relativt lite XML analysebibliotek. Den kreves av Perl modulen XML::Parser.

- Expect

  Denne pakken inneholder et program for å utføre skriptete dialoger med andre interaktive programmer. Det er ofte brukt for testing av andre pakker.

- File

  Denne pakken inneholder et verktøy for å bestemme typen av en gitt fil eller filer. Noen få pakker trenger det i byggeskriptene deres.

- Findutils

  Denne pakken inneholder programmer for å finne filer i et filsystem. Det brukes i mange pakkers byggeskript.

- Flex

  Denne pakken inneholder et verktøy for å generere programmer som gjenkjenne mønstre i tekst. Det er GNU versjonen av lex (leksikalsk analysator) programmet. Det kreves for å bygge flere LFS pakker.

- Gawk

  Denne pakken inneholder programmer for å manipulere tekstfiler. Det er GNU versjonen av awk (Aho-Weinberg-Kernighan). Den brukes i mange andre pakkers byggeskript.

- GCC

  Denne pakken er Gnu Kompilatorsamlingen. Den inneholder C og C++ kompilatorer samt flere andre som ikke er bygget av LFS.

- GDBM

  Denne pakken inneholder GNU Database behandlings biblioteket. Den brukes av en annen LFS pakke, Man-DB.

- Gettext

  Denne pakken inneholder verktøy og biblioteker for internasjonalisering og lokalisering av en rekke pakker.

- Glibc

  Denne pakken inneholder C hovedbiblioteket. Linux programmer vil ikke kjøre uten.

- GMP

  Denne pakken inneholder matematiske biblioteker som gir nyttige funksjoner for vilkårlig presisjonsaritmetikk. Det kreves for å bygge GCC.

- Gperf

  Denne pakken inneholder et program som genererer en perfekt hash funksjon fra et nøkkelsett. Det kreves for Eudev.

- Grep

  Denne pakken inneholder programmer for å søke gjennom filer. Disse programmene brukes av de fleste pakkenes byggeskript.

- Groff

Denne pakken inneholder programmer for behandling og formatering av tekst. En viktig funksjon av disse programmene er å formatere man sider.

- GRUB

  Denne pakken er Grand Unified Boot Loader. Det er en av flere tilgjengelige oppstartslastere, men er den mest fleksible.

- Gzip

  Denne pakken inneholder programmer for komprimering og dekomprimere av filer. Det er nødvendig for å dekomprimere mange pakker i LFS og utover.

- Iana-etc

  Denne pakken gir data for nettverkstjenester og protokoller. Det er nødvendig for å aktivere riktige nettverksfunksjoner.

- Inetutils

  Denne pakken inneholder programmer for grunnleggende nettverksadministrasjon.

- Intltool

  Denne pakken inneholder verktøy for å trekke ut oversettbare strenger fra kildefiler.

- IProute2

  Denne pakken inneholder programmer for grunnleggende og avansert IPv4 og IPv6 nettverk. Det ble valgt fremfor det andre felles nettverks verktøypakke (net-tools) for sine IPv6-funksjoner.

- Jinja2

  Denne pakken er en Python modul for å lage tekstmaler. Det kreves for å bygge Systemd.

- Kbd

  Denne pakken inneholder nøkkeltabellfiler, tastaturverktøy for ikke-amerikanske tastaturer, og en rekke konsollfonter.

- Kmod

  Denne pakken inneholder programmer som trengs for å administrere Linux kjernemoduler.

- Less

  Denne pakken inneholder en veldig fin tekstfilviser som lar deg rulle opp eller ned når du viser en fil. Den brukes også av Man-DB for visning av man sider.

- Libcap

  Denne pakken implementerer brukerromsgrensesnittene til POSIX 1003.1e funksjonene tilgjengelig i Linux kjerner.

- Libelf

  Elfutils prosjektet gir biblioteker og verktøy for ELF filer og DWARF data. De fleste verktøyene i denne pakken er tilgjengelige i andre pakker, men biblioteket er nødvendig for å bygge Linux kjernen som bruker standard (og mest effektive) konfigurasjon.

- Libffi

Denne pakken implementerer et grensesnitt for overførbart programmering på høyt nivå til ulike kallkonvensjoner. Noen programmer vet kanskje ikke på sammenstillingstidspunktet hvilke argumenter som skal overføres til en funksjon. For eksempel kan en tolk bli fortalt under kjøringen om antallet og typene argumenter som brukes til å kalle en gitt funksjon. Libffi kan brukes i slike programmer for å gi en bro fra tolkeprogrammet til kompilert kode.

- Libpipeline

  Libpipeline pakken inneholder et bibliotek for å manipulere kommandokøer av delprosesser på en fleksibel og praktisk måte. Det kreves av Man-DB pakken.

- Libtool

  Denne pakken inneholder GNU generiske bibliotekstøtte skript. Det omslutter kompleksiteten ved å bruke delte biblioteker i en konsekvent, bærbart grensesnitt. Det trengs av testpakker i andre LFS pakker.

- Linux Kernel

  Denne pakken er operativsystemet. Det er Linux i GNU/Linux miljøet.

- M4

  Denne pakken inneholder en generell tekstmakroprosessor som er nyttig som byggeverktøy for andre programmer.

- Make

  Denne pakken inneholder et program for å styre byggingen av pakker. Det kreves av nesten alle pakker i LFS.

- MarkupSafe

  Denne pakken er en Python modul for å behandle strenger i HTML/XHTML/XML trygt. Jinja2 krever denne pakken.

- Man-DB

  Denne pakken inneholder programmer for å finne og vise man sider. Det ble valgt i stedet for man pakken på grunn av overlegne internasjonaliseringsevner. Det leverer man programmet.

- Man-pages

  Denne pakken inneholder det faktiske innholdet i det grunnleggende Linux man sider.

- Meson

  Denne pakken inneholder et programvareverktøy for å automatisere byggingen av programvare. Hovedmålet for Meson er å minimere tiden som programvareutviklere må bruke på å konfigurere bygge systemet. Det kreves for å bygge Systemd, så vel som mange BLFS pakker.

- MPC

  Denne pakken inneholder funksjoner for aritmetikk av komplekse tall. Det kreves av GCC.

- MPFR

  Denne pakken inneholder funksjoner for multiple presisjons aritmetikk. Det kreves av GCC.

- Ninja

  Denne pakken inneholder et lite byggesystem med fokus på hastighet. Den er designet for å ha inndatafilene generert på høyere nivå av et bygge system, og å kjøre bygget så raskt som mulig. Denne pakken kreves av Meson.

- Ncurses

Denne pakken inneholder biblioteker for terminaluavhengig håndtering av skjermkarakterer. Det brukes ofte til å gi markørkontroll for et menysystem. Det trengs av en rekke pakker i LFS.

- Openssl

  Denne pakken inneholder administrasjonsverktøy og biblioteker knyttet til kryptografi. Disse er nyttige for å gi kryptografiske funksjoner til andre pakker, inkludert Linux kjernen.

- Patch

  Denne pakken inneholder et program for å endre eller lage filer ved å bruke en *oppdateringsfil* vanligvis opprettet av diff programmet. Det trengs av byggeprosedyren for flere LFS pakker.

- Perl

  Denne pakken er en tolk for kjøretidsspråket PERL. Det er nødvendig for installasjon og testpakker for flere LFS pakker.

- Pkg-config

  Denne pakken gir et program som returnerer metadata om en installert bibliotek eller pakke.

- Procps-NG

  Denne pakken inneholder programmer for overvåking av prosesser. Disse programmer er nyttige for systemadministrasjon, og brukes også av LFS Oppstartsskript.

- Psmisc

  Denne pakken inneholder programmer for å vise informasjon om prosesser som kjører. Disse programmene er nyttige for system administrasjon.

- Python 3

  Denne pakken gir et tolkeprogram som har en design filosofi som legger vekt på kodelesbarhet.

- Readline

  Denne pakken er et sett med biblioteker som tilbyr redigerings- og historikkfunksjoner på kommandolinjen. Den brukes av Bash.

- Sed

  Denne pakken tillater redigering av tekst uten å åpne den i en tekstredigerer. Det er også nødvendig for de fleste LFS pakkers konfigureringsskript.

- Shadow

  Denne pakken inneholder programmer for håndtering av passord på en sikker måte.

- Systemd

  Denne pakken gir et init program og flere ekstra oppstarts- og systemkontrollfunksjoner som et alternativ til Sysvinit. Den brukes av mange kommersielle distribusjoner.

- Tar

  Denne pakken gir arkiverings- og utpakkingssmuligheter av praktisk talt alle pakker som brukes i LFS.

- Tcl

  Denne pakken inneholder Verktøykommandospråk som brukes i mange testpakker i LFS pakker.

- Texinfo

  Denne pakken inneholder programmer for lesing, skriving og konvertere informasjonssider. Den brukes i installasjons prosedyrer for mange LFS pakker.

- Util-linux

  Denne pakken inneholder diverse hjelpeprogrammer. Blant dem er verktøy for håndtering av filsystemer, konsoller, partisjoner og meldinger.

- Vim

  Denne pakken inneholder et redigeringsprogram. Den ble valgt på grunn av sin kompatibilitet med det klassiske vi redigeringsprogrammet og dens enorme antall kraftige kapasiteter.. Et redigeringsprogram er et veldig personlig valg for mange brukere og andre redigeringsprogram kan brukes om ønskelig.

- XML::Parser

  Denne pakken er en Perl modul som har grensesnitt med Expat.

- XZ Utils

  Denne pakken inneholder programmer for komprimering og dekomprimering av filer. Det gir den høyeste kompresjonen som generelt er tilgjengelig og er nyttig for å dekomprimere pakker i XZ- eller LZMA-format.

- Zlib

  Denne pakken inneholder komprimerings- og dekompresjonsrutiner som brukes av noen programmer.

- Zstd

  Denne pakken inneholder komprimerings- og dekompresjonsrutiner som brukes av noen programmer. Det gir høyt kompresjonsforhold og en svært bredt utvalg av kompresjon/hastighets avveininger.

Typografi

For å gjøre ting lettere å følge, er det noen få typografiske konvensjoner brukt gjennom denne boken. Denne delen inneholder noen eksempler på det typografiske formatet som finnes i hele Linux From Scratch.

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

Denne formen for tekst er designet for å skrives nøyaktig slik den er skrevet med mindre noe annet er notert i den omkringliggende teksten. Det brukes også i forklaringsseksjoner for å identifisere hvilke av kommandoene det refereres til.

I noen tilfeller utvides en logisk linje til to eller flere fysiske linjer med en omvendt skråstrek på slutten av linjen.

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

Merk at omvendt skråstrek må følges av en umiddelbar retur. Annen mellomromstegn som mellomrom eller tabulatortegn vil lage feil resultater.

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

Denne formen for tekst (tekst med fast bredde) viser skjermutdata, vanligvis som resultatet av utstedte kommandoer. Dette formatet brukes også til å vise filnavn, som for eksempel `/etc/ld.so.conf`.

Uthevet

Denne tekstformen brukes til flere formål i boken. Dens viktigste formålet er å understreke viktige punkter eller elementer.

https://www.linuxfromscratch.org/

Dette formatet brukes for hyperkoblinger både innenfor LFS-fellesskapet og til eksterne sider. Det inkluderer HOWTOer, nedlastingssteder og nettsteder.

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

Dette formatet brukes når du oppretter konfigurasjonsfiler. Den første kommandoen ber systemet lage filen `$LFS/etc/group` fra hva som enn skrives på de følgende linjene til sekvensen End Of File (EOF) er påtruffet. Derfor er hele denne delen vanligvis skrevet som det vises.

<REPLACED TEXT>

Dette formatet brukes til å kapsle inn tekst som ikke skal skrives som det vises eller for kopier-og-lim-operasjoner.

[VALGFRI TEKST]

Dette formatet brukes til å kapsle inn tekst som er valgfri.

`passwd(5)`

Dette formatet brukes til å referere til en spesifikk manual (man) side. Tallet innenfor parentes indikerer en bestemt del i manualene. For eksempel, **passwd** har to man sider. I henhold til LFS installasjonsinstruksjoner, disse to man sidene vil være plassert på `/usr/share/man/man1/passwd.1` og `/usr/share/man/man5/passwd.5`. Når boken bruker `passwd(5)` refererer den spesifikt refererer til `/usr/share/man/man5/passwd.5`. **man passwd** vil skrive ut den første man siden den finner som stemmer med "passwd", som vil bli `/usr/share/man/man1/passwd.1`. For dette eksemplet må du kjøre **man 5 passwd** for å lese siden som blir spesifisert. Merk at de fleste man sider ikke har duplikate sidenavn i forskjellige seksjoner. Derfor, **man *<programmet name>*** er generelt tilstrekkelig.

Struktur

Denne boken er delt inn i følgende deler.

Del I - Introduksjon

Del I forklarer noen viktige merknader om hvordan du går frem med LFS installasjon. Denne delen gir også metainformasjon om boken.

Del II - Forberedelse til bygging

Del II beskriver hvordan du forbereder byggeprosessen—lage en partisjon, nedlasting av pakkene og kompilering av midlertidige verktøy.

Del III - Bygging av LFS kryssverktøykjede og midlertidige verktøy

Del III gir instruksjoner for å bygge verktøyene nødvendig for å konstruere det endelige LFS systemet.

Del IV - Bygge LFS systemet

Del IV guider leseren gjennom byggingen av LFS systemet—kompilere og installere alle pakkene én etter én, sette opp oppstartsskriptene og installere kjernen. Det resulterende Linux-systemet er grunnlaget som annen programvare kan bygges på, utvide systemet etter ønske. På slutten av denne boken er det en enkel å bruke referanse som viser alle programmene, bibliotekene og viktige filer som er installert.

Del V - Vedlegg

Del V gir informasjon om selve boken inkludert akronymer og termer, anerkjennelser, pakkeavhengigheter, en liste over LFS-oppstartsskript, lisenser for distribusjon av bok, og en omfattende indeks over pakker, programmer, biblioteker, og skript.

Errata og sikkerhetsråd

Programvaren som brukes til å lage et LFS system blir kontinuerlig oppdatert og forbedret. Sikkerhetsadvarsler og feilrettinger kan bli tilgjengelige etter at LFS boken er utgitt. For å sjekke om pakkeversjonene eller instruksjonene i denne utgaven av LFS trenger eventuelle modifikasjoner for å imøtekomme sikkerhetssårbarheter eller andre feilrettinger, vennligst besøk *https://www.linuxfromscratch.org/lfs/errata/11.1-systemd/* før du fortsetter med byggingen. Du bør merke noen endringer som vises, og bruke dem på den relevante delen av boken mens du bygger LFS systemetet.

I tillegg opprettholder Linux From Scratch redaktørene en liste over sikkerhetssårbarheter oppdaget etter at en bok ble utgitt. For å sjekke om det er noen aktive sikkerhetssårbarheter, vennligst besøk *https://www.linuxfromscratch.org/lfs/advisories/* før du fortsetter med byggingen. Du bør merke deg eventuelle råd og utfør trinnene for å fikse eventuelle sikkerhetssårbarheter mens du bygger LFS systemet.

Part I. Introduksjon

Chapter 1. Introduksjon

1.1. Hvordan bygge et LFS-system

LFS-systemet vil bli bygget ved å bruke en allerede installert Linux-distribusjon (som Debian, OpenMandriva, Fedora eller openSUSE). Dette eksisterende Linux-system (verten) vil bli brukt som utgangspunkt for gi nødvendige programmer, inkludert en kompilator, linker og skall, å bygge det nye systemet. Velg “development” alternativ under distribusjonsinstallasjonen for å kunne få tilgang til disse verktøy.

Som et alternativ til å installere en separat distribusjon på din maskinen, du kanskje ønsker å bruke en LiveCD fra en kommersiell distribusjon.

Kapittel 2 i denne boken beskriver hvordan lage en ny Linuxpartisjon og et nytt filsystem. Dette er stedet hvor det nye LFS systemet skal kompileres og installeres. Kapittel 3 forklarer hvilke pakker og oppdateringer som må lastes ned for å bygge et LFS system og hvordan lagre dem på det nye filsystemet. Kapittel 4 diskuterer oppsettet av et hensiktsmessig arbeidsmiljø. Vennligst les Kapittel 4 nøye som det forklarer flere viktige problemer du må være klar over før du begynner å jobbe deg gjennom Kapittel 5 og utover.

Kapittel 5, forklarer installasjonen av den første verktøykjeden (binutils, gcc og glibc) ved bruk av krysskompilerings teknikker for å isolere de nye verktøyene fra vertssystemet.

Kapittel 6 viser hvordan du krysskompiler grunnleggende verktøy ved å bruke den nettopp bygde kryssverktøykjeden.

Kapittel 7 går deretter inn i et "chroot" miljø og bruker de tidligere bygde verktøyene til å bygge tilleggsverktøyene som trengs for å bygge og teste det endelige systemet.

Denne innsatsen for å isolere det nye systemet fra vertsdistribusjonen kan virke overdreven. En fullstendig teknisk forklaring på hvorfor dette gjøres er gitt inn Toolchain Technical Notes.

I Chapter 8, Det fulle LFS system blir bygget. En annen fordel gitt av chroot miljøet er at det lar deg fortsette å bruke vertssystemet mens LFS bygges. Mens du venter på at pakkesammenstillinger blir fullført, kan du fortsette å bruke datamaskinen som normalt.

For å fullføre installasjonen er den grunnleggende systemkonfigurasjonen satt opp i Kapittel 9, og kjernen og oppstartslasteren er satt opp i Kapittel 10. Kapittel 11 inneholder informasjon om å fortsette LFS opplevelsen utover denne boken. Etter at trinnene i denne boken er implementert, vil datamaskinen være klar til å starte på nytt i det nye LFS systemet.

Dette er prosessen i et nøtteskall. Detaljert informasjon om hvert trinn er diskutert i de følgende kapitlene og pakkebeskrivelsene. Punkter som kan virke kompliserte vil bli avklart, og alt vil falle på plass når du legger ut på LFS eventyret.

1.2. Hva er nytt siden forrige utgivelse

I denne versjonen av LFS har det skjedd en større omorganisering av boken ved å bruke teknikker som unngår å endre vertssystemet og gir en mer rett frem byggeprosess.

Nedenfor er en liste over pakkeoppdateringer gjort siden forrige utgivelse av boken.

Oppgradert til:

-
- Automake-1.16.5
- Bash 5.1.16

- Bc 5.2.2
- Binutils-2.38
- Bison-3.8.2
- Coreutils-9.0
- E2fsprogs-1.46.5
- Expat-2.4.6
- File-5.41
- Findutils-4.9.0
- Gawk-5.1.1
- GDBM-1.23
- Glibc-2.35
- Gzip-1.11
- IANA-Etc-20220207
- Inetutils-2.2
- IPRoute2-5.16.0
- Jinja2-3.0.3
- Libcap-2.63
- Libelf-0.186 (from elfutils)
- Libpipeline-1.5.5
- Linux-5.16.9
- Man-DB-2.10.1
- Meson-0.61.1
- Ncurses-6.3
- Openssl-3.0.1
- Python-3.10.2
- Readline-8.1.2
- Shadow-4.11.1
- Systemd-250
- Tcl-8.6.12
- Tzdata-2021e
- Util-Linux-2.37.4
- Vim-8.2.4383
- Zstd-1.5.2

Lagt til:

-
- binutils-2.38-lto_fix-1.patch
- coreutils-9.0-chmod_fix-1.patch
- file-5.40-upstream_fixes-1.patch
- shadow-4.10-useradd_segfault-1.patch
- systemd-250-upstream_fixes-1.patch

Fjernet:

•

1.3. Endringslogg

Dette er versjon 11.1-systemd av Linux From Scratch-boken, datert 1. Mars 2022. Hvis denne boken er mer enn seks måneder gammel, en nyere og bedre versjonen er sannsynligvis allerede tilgjengelig. For å finne ut, vennligst sjekk et av speilene via *https://www.linuxfromscratch.org/mirrors.html*.

Nedenfor er en liste over endringer som er gjort siden forrige utgivelse av boken.

Endringsloggoppføringer:

- 01.03.2022
  - [bdubbs] - LFS-11.1 utgitt.
- 23.02.2022
  - [bdubbs] - Oppdatert til expat-2.4.6 (sikkerhetsretting). Rettinger *#5011*.
- 15.02.2022
  - [bdubbs] - LFS-11.1-rc1 utgitt.
  - [bdubbs] - La til binutils-2.38 LTO oppdatering. Rettinger *#5011*.
  - [bdubbs] - Oppdatert til util-linux-2.37.4. Rettinger *#5010*.
  - [bdubbs] - Oppdatert til man-db-2.10.1. Rettinger *#5009*.
  - [bdubbs] - Oppdatert til linux-5.16.9. Rettinger *#5008*.
  - [bdubbs] - Oppdatert til vim-8.2.4383 (Sikkerhetsrettinger). Adresserer *#4500*.
  - [bdubbs] - Oppdatert til iana-etc-20220207. Adresserer *#5006*.
- 10.02.2022
  - [xry111] - Omgå problemet som forårsaker at binærfiler lenker til biblioteker fra vertsdistroen for pass 2 binutils. Nå er det unødvendig å bygge zlib i kapittel 6.
- 09.02.2022
  - [bdubbs] - Oppdatert til bc-5.2.2. Rettinger *#5004*.
  - [bdubbs] - Oppdatert til linux-5.16.8. Rettinger *#5005*.
  - [bdubbs] - Oppdatert til binutils-2.38. Krever at zlib legges til i kapittel 6. Rettinger *#5007*.
- 04.02.2022
  - [xry111] - Fjerne **bash** +h direktiver i chroot. Rettinger *#4998*.
  - [xry111] - Oppdatert til man-db-2.10.0. Rettinger *#5002*.
  - [xry111] - Flytt OpenSSL før Kmod og aktiver OpenSSL for Kmod bygge.
  - [xry111] - Oppdatert til gdbm-1.23. Rettinger *#5000*.
  - [xry111] - Oppdatert til tcl-8.6.12. Rettinger *#5001*.
  - [thomas] - Fjerne sed fra glibc-instruksjonene i kapittel 8. Den har blitt anvendt oppstrøms.
- 03.02.2022
  - [bdubbs] - La til coreutils-9.0 chmod oppdatering. Rettinger *#4992*.

- [bdubbs] - Oppdatert til glibc-2.35. Rettinger *#4999*.
- [bdubbs] - Oppdatert til linux-5.16.5. Rettinger *#4996*.
- [bdubbs] - Oppdatert til findutils-4.9.0. Rettinger *#4995*.
- [bdubbs] - Oppdatert til expat-2.4.4. Rettinger *#4993*.
- [bdubbs] - Oppdatert til iana-etc-20220128. Rettinger *#4994*.

- 29.01.2022
  - [bdubbs] - Oppdatert til linux-5.16.4. Rettinger *#4991*.
- 27.01.2022
  - [bdubbs] - Oppdatert til vim-8.2.4236. Adresserer *#4500*.
  - [bdubbs] - Oppdatert til zstd-1.5.2. Rettinger *#4988*.
  - [bdubbs] - Oppdatert til util-linux-2.37.3 (sikkerhetsretting). Rettinger *#4989*.
  - [bdubbs] - Oppdatert til Python-3.10.2. Rettinger *#4987*.
  - [bdubbs] - Oppdatert til linux-5.16.2. Rettinger *#4979*.
  - [bdubbs] - Oppdatert til libcap-2.63. Rettinger *#4990*.
  - [bdubbs] - Oppdatert til iproute2-5.16.0. Rettinger *#4982*.
  - [bdubbs] - Oppdatert til iana-etc-20220120. Rettinger *#4975*.
- 20.01.2022
  - [bdubbs] - Oppdatert til expat-2.4.3 (sikkerhetsrettinger). Rettinger *#4984*.
  - [pierre] - Oppdatert til meson-0.61.1. Rettinger *#4985*.
- 17.01.2022
  - [thomas] - Lagt til en rettelse av en skrivefeil i meson-0.61.0 oppdateringen.
- 15.01.2022
  - [bdubbs] - Oppdatert til shadow-4.11.1. Rettinger *#4976*.
  - [bdubbs] - Oppdatert til readline-8.1.2. Rettinger *#4980*.
  - [bdubbs] - Oppdatert til meson-0.61.0. Rettinger *#4983*.
  - [bdubbs] - Oppdatert til libpipeline-1.5.5. Rettinger *#4977*.
  - [bdubbs] - Oppdatert til bash-5.1.16. Rettinger *#4978*.
- 13.01.2021
  - [renodr] - Fikset CVE-2021-3997 i systemd, samt fiksing av et problem med standard vertsnavn og inaktive enheter. Rettinger *#4981*.
- 03.01.2021
  - [renodr] - La til ytterligere kjernekonfigurasjon for å tillate 'systemd-oomd' å fungere.
- 01.01.2022
  - [bdubbs] - Oppdatert til e2fsprogs-1.46.5. Rettinger *#4974*.
  - [bdubbs] - Oppdatert til zstd-1.5.1. Rettinger *#4972*.
  - [bdubbs] - Oppdatert til expat-2.4.2. Rettinger *#4970*.

- [bdubbs] - Oppdatert til shadow-4.10. Rettinger *#4969*.
- [bdubbs] - Oppdatert til linux-5.15.12. Rettinger *#4967*.
- [bdubbs] - Oppdatert til iana-etc-20211224. Rettinger *#4962*.
- [bdubbs] - Oppdatert til openssl-3.0.1. Rettinger *#4922*.
- [bdubbs] - Oppdatert til eudev-3.2.11. Rettinger *#4914*.

- 30.12.2021
  - [renodr] - Oppdatert til systemd-250. Rettinger *#4971*.
  - [renodr] - Oppdatert til meson-0.60.3. Rettinger *#4973*.
- 15.12.2021
  - [bdubbs] - Oppdatert til python3-3.10.1. Rettinger *#4963*.
  - [bdubbs] - Oppdatert til openssl-1.1.1m. Rettinger *#4966*.
  - [bdubbs] - Oppdatert til linux-5.15.7. Rettinger *#4964*.
  - [bdubbs] - Oppdatert til libcap-2.62. Rettinger *#4965*.
- 14.12.2021
  - [thomas] - Tillat å bygge findutils på 32-biters systemer. Valgt fra grenen multilib av [pierre].
- 01.12.2021
  - [bdubbs] - Oppdatert til vim-8.2.3704. Adresserer *#4500*.
  - [bdubbs] - Oppdatert til iana-etc-20211124. Rettinger *#4957*.
  - [bdubbs] - Oppdatert til bc-5.2.1. Rettinger *#4959*.
  - [bdubbs] - Oppdatert til meson-0.60.2. Rettinger *#4960*.
  - [bdubbs] - Oppdatert til linux-5.15.5. Rettinger *#4956*.
- 15.11.2021
  - [bdubbs] - Oppdatert til iana-etc-20211112. Rettinger *#4955*.
  - [bdubbs] - Oppdatert til elfutils-0.186. Rettinger *#4954*.
  - [bdubbs] - Oppdatert til jinja2-3.0.3. Rettinger *#4953*.
  - [bdubbs] - Oppdatert til bc-5.2.0. Rettinger *#4952*.
  - [bdubbs] - Oppdatert til ncurses-6.3. Rettinger *#4951*.
  - [bdubbs] - Oppdatert til libpipeline-1.5.4. Rettinger *#4950*.
  - [bdubbs] - Oppdatert til meson-0.60.1. Rettinger *#4949*.
  - [bdubbs] - Oppdatert til iproute2-5.15.0. Rettinger *#4948*.
  - [bdubbs] - Oppdatert til linux-5.15.2. Rettinger *#4947*.
- 01.11.2021
  - [bdubbs] - Oppdatert til gawk-5.1.1. Rettinger *#4946*.
  - [bdubbs] - Oppdatert til meson-0.60.0. Rettinger *#4945*.
  - [bdubbs] - Oppdatert til libcap-2.60. Rettinger *#4944*.
  - [bdubbs] - Oppdatert til gdbm-1.22. Rettinger *#4943*.

- [bdubbs] - Oppdatert til file-5.41. Rettinger *#4942*.
- [bdubbs] - Oppdatert til linux-5.14.15. Rettinger *#4941*.
- [bdubbs] - Oppdatert til iana-etc-20211025. Rettinger *#4940*.
- [bdubbs] - Oppdatert til tzdata-2021e. Rettinger *#4939*.

- 15.10.2021
  - [bdubbs] - Oppdatert til vim-8.2.3508. Adresserer *#4500*.
  - [bdubbs] - Oppdatert til tzdata-2021c. Rettinger *#4934*.
  - [bdubbs] - Oppdatert til Python-3.10.0. Rettinger *#4938*.
  - [bdubbs] - Oppdatert til Jinja2-3.0.2. Rettinger *#4937*.
  - [bdubbs] - Oppdatert til linux-5.14.12. Rettinger *#4932*.
  - [bdubbs] - Oppdatert til iana-etc-20211004. Rettinger *#4933*.
  - [bdubbs] - Oppdatert til bc-5.1.1. Rettinger *#4936*.
  - [bdubbs] - Oppdatert til automake-1.16.5. Rettinger *#4935*.
- 01.10.2021
  - [bdubbs] - Oppdatert til vim-8.2.3458. Adresserer *#4500*.
  - [bdubbs] - Oppdatert til iana-etc-20210924. Adresserer *#4722*.
  - [bdubbs] - Oppdatert til tzdata-2021b. Rettinger *#4929*.
  - [bdubbs] - Oppdatert til meson-0.59.2. Rettinger *#4931*.
  - [bdubbs] - Oppdatert til linux-5.14.8. Rettinger *#4925*.
  - [bdubbs] - Oppdatert til libcap-2.59. Rettinger *#4926*.
  - [bdubbs] - Oppdatert til coreutils-9.0. Rettinger *#4928*.
  - [bdubbs] - Oppdatert til bison-3.8.2. Rettinger *#4930*.
- 15.09.2021
  - [bdubbs] - Sørget for at instruksjoner for tcl dokumentasjon er tilstede. Rettinger *#4923*.
  - [bdubbs] - Oppdatert til Python3-3.9.7. Rettinger *#4916*.
  - [bdubbs] - Oppdatert til linux-5.14.3. Rettinger *#4913*.
  - [bdubbs] - Oppdatert til libcap-2.57. Rettinger *#4912*.
  - [bdubbs] - Oppdatert til iproute2-5.14.0. Rettinger *#4917*.
  - [bdubbs] - Oppdatert til inetutils-2.2. Rettinger *#4918*.
  - [bdubbs] - Oppdatert til gzip-1.11. Rettinger *#4920*.
  - [bdubbs] - Oppdatert til gdbm-1.21. Rettinger *#4919*.
  - [bdubbs] - Oppdatert til bison-3.8.1. Rettinger *#4921*.
  - [bdubbs] - Oppdatert til bc-5.0.2. Rettinger *#4905*.
- 08.09.2021
  - [renodr] - Fiks regresjoner i File som resulterer i feil gjenkjenning av tekst og XZ filer.
- 06.09.2021

    - [bdubbs] - Tekstforklaringer i delen for sikkerhetskopiering/gjenoppretting av kapittel 7. Takk til Kevin Buckley for oppdateringen.
- 01.09.2021
    - [bdubbs] - LFS-11.0 utgitt.

1.4. Ressurser

1.4.1. FAQ

Hvis du under byggingen av LFS systemet støter på noen feil, har spørsmål eller tror det er en skrivefeil i boken, vennligst start med å se de vanlige spørsmålene (FAQ) som befinner seg på *https://www.linuxfromscratch.org/faq/*.

1.4.2. E-postlister

`linuxfromscratch.org` serveren er vert for en rekke E-post lister brukt til utvikling av LFS prosjektet. Disse listene inkluderer hovedutviklings- og støttelister, blant annet. Hvis FAQ ikke løser problemet du har, vil neste trinn være å søke i E-post listene på *https://www.linuxfromscratch.org/search.html*.

For informasjon om de forskjellige listene, hvordan abonnere, arkiv steder og tilleggsinformasjon, besøk *https://www.linuxfromscratch.org/mail.html*.

1.4.3. IRC

Flere medlemmer av LFS fellesskapet tilbyr assistanse på Internett Relay Chat (IRC). Før du bruker denne støtten, sørg for at dine spørsmål ikke allerede er besvart i LFS FAQ eller E-postlistenens arkiv. Du finner IRC-nettverket på `irc.libera.chat`. Støttekanalen heter #lfs-support.

1.4.4. Speilnettsteder

LFS prosjektet har en rekke verdensomspennende speil for å få tilgang til nettstedet og laste ned de nødvendige pakkene mer praktisk. Vær så snill besøk LFS nettstedet på *https://www.linuxfromscratch.org/mirrors.html* for en liste av nåværende speil.

1.4.5. Kontaktinformasjon

Send alle dine spørsmål og kommentarer til en av LFS E-postlister (se ovenfor).

1.5. Hjelp

Hvis det oppstår et problem eller et spørsmål mens du arbeider gjennom denne boken, vennligst sjekk siden FAQ på *https://www.linuxfromscratch.org/faq/#generalfaq*. Spørsmål er ofte allerede besvart der. Hvis spørsmålet ditt ikke er svart på denne siden, prøv å finne kilden til problemet. De følgende tips vil gi deg veiledning for feilsøking: *https://www.linuxfromscratch.org/hints/downloads/files/errors.txt*.

Hvis du ikke finner problemet oppført i vanlige spørsmål, søk i E-post listene på *https://www.linuxfromscratch.org/search.html*.

Vi har også et fantastisk LFS fellesskap som er villig til å tilby hjelp gjennom E-postlistene og IRC (se Section 1.4, “Ressurser” delen av denne boken). Imidlertid får vi flere brukerspørsmål hver dag, og mange av dem kan være besvart gjennom FAQ og gjennom E-postlistene, søk der først. Så for at vi skal kunne tilby best mulig hjelp, må du gjøre noe forskning på egen hånd først. Det lar oss fokusere på de mere uvanlige brukerstøtte. Hvis søkene dine ikke gir en løsning, vennligst ta med all relevant informasjon (nevnt nedenfor) i din forespørsel om hjelp.

1.5.1. Ting å nevne

Bortsett fra en kort forklaring av problemet som oppleves, de viktigste tingene å inkludere i enhver forespørsel om hjelp er:

- Versjonen av boken som brukes (i dette tilfellet 11.1-systemd)
- Vertsdistribusjonen og versjonen som brukes til å lage LFS
- Utdata fra Systemkrav for vert skriptet
- Pakken eller seksjonen problemet ble oppdaget i
- Den nøyaktige feilmeldingen eller symptomet som mottas
- Gi beskjed om du har avviket fra boken

Note

Avvik fra denne boken gjør *ikke* at vi ikke vil hjelpe deg. Tross alt handler LFS om personlig preferanse. Å være på forhånd om eventuelle endringer i den etablerte prosedyren hjelper oss å vurdere og finne mulige årsaker til problemet ditt.

1.5.2. Konfigurasjonsskript problemer

Hvis noe går galt mens du kjører **configure** skript, gjennomgå `config.log` filen. Denne filen kan inneholde feil oppstått under **configure** som ikke ble skrevet ut på skjermen. Inkluder *relevante* linjer hvis du trenger å be om hjelp.

1.5.3. Kompileringsproblemer

Både skjermutdata og innholdet i ulike filer er nyttige ved å fastslå årsaken til kompileringsproblemer. Skjermens utdata fra **configure** skriptet og **make** kjøringen kan være nyttig. Det er ikke nødvendig å inkludere hele utdataen, men inkludere nok av relevant informasjon. Nedenfor er et eksempel på type informasjon som skal inkluderes fra skjermens utdata fra **make**:

```
gcc -DALIASPATH=\"/mnt/lfs/usr/share/locale:.\"
-DLOCALEDIR=\"/mnt/lfs/usr/share/locale\"
-DLIBDIR=\"/mnt/lfs/usr/lib\"
-DINCLUDEDIR=\"/mnt/lfs/usr/include\" -DHAVE_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 signame.o variable.o vpath.o
default.o remote-stub.o version.o opt1.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
```

I dette tilfellet vil mange mennesker bare inkludere seksjonen fra bunnen:

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

Dette er ikke nok informasjon til å diagnostisere problemet riktig fordi den bare merker at noe gikk galt, ikke *hva* som gikk galt. Hele delen, som i eksempelet ovenfor, er det som skal lagres fordi det inkluderer kommandoen som ble utført og tilhørende feilmelding(er).

En utmerket artikkel om å be om hjelp på Internett er tilgjengelig på nett på *http://catb.org/~esr/faqs/smart-questions.html*. Les og følg tipsene i dette dokumentet for å øke sannsynligheten for å få hjelpen du trenger.

Part II. Forbereder for byggingen

Chapter 2. Klargjøring av vertssystemet

2.1. Introduksjon

I dette kapittelet, vertsverktøyene som trengs for å bygge LFS kontrolleres og om nødvendig installeres. Deretter vil en partisjon klargjøres som vert for LFS systemet. Vi lager partisjonen selv, lager et filsystem på den og monter den.

2.2. Systemkrav for vert

Vertssystemet ditt bør ha følgende programvare med minimumsversjoner angitt. Dette burde ikke være et problem for de fleste moderne Linux-distribusjoner. Vær også oppmerksom på at mange distribusjoner vil plassere programvaredeklarasjoner i separate pakker, ofte i form av "<package-name>-devel" eller "<package-name>-dev". Pass på å installere disse hvis distribusjonen din gir dem.

Tidligere versjoner av de oppførte programvarepakkene kan fungere, men har ikke blitt testet.

- **Bash-3.2** (/bin/sh bør være en symbolsk eller hard lenke til bash)
- **Binutils-2.13.1** (Versjoner større enn 2.38 anbefales ikke ettersom de ikke har blitt testet)
- **Bison-2.7** (/usr/bin/yacc bør være en lenke til bison eller et lite skript som kjører bison)
- **Coreutils-6.9**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk bør være en link til gawk)
- **GCC-4.8** inkludert C++ kompilatoren, **g++** (Versjoner større enn 11.2.0 er ikke anbefalt da de ikke er testet). C og C++ standard biblioteker (med deklarasjoner) må også være tilstede slik at C++ kompilatoren kan bygge vertsbaserte programmer
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Linux Kernel-3.2**

  Grunnen til kravet om kjerneversjon er at vi spesifiserer den versjonen når du bygger glibc i Kapittel 5 og Chapter 8, etter anbefaling fra utviklerne. Det kreves også av udev.

  Hvis vertskjernen er tidligere enn 3.2 du må erstatte kjernen med en mer oppdatert versjon. Det er to måter du kan gjøre dette på. Først, se om din Linux leverandør tilbyr en 3.2 eller senere kjernepakke. I så fall kan det være lurt å installere den. Hvis din leverandøren ikke tilbyr en akseptabel kjernepakke, eller du foretrekker å la være å installer den, kan du kompilere en kjerne selv. Instruksjoner for å kompilere kjernen og konfigurere oppstartslasteren (forutsatt at verten bruker GRUB) er lokalisert i Kapittel 10.
- **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-4.7**
- **Xz-5.0.0**

Important

Merk at symbollenkene nevnt ovenfor er nødvendige for å bygge et LFS system ved å bruke instruksjonene i denne boken. Symlinker som peker på annen programvare (som dash, mawk osv.) kan fungere, men er ikke testet eller støttet av LFS-utviklingsteamet, og kan kreve enten avvik fra instruksjonene eller tilleggsoppdateringer til noen pakker.

For å se om vertssystemet ditt har alle de riktige versjonene, og muligheten til å kompilere programmer, kjør følgende:

```
cat > version-check.sh << "EOF"
#!/bin/bash
# Simple script to list version numbers of critical development tools
export LC_ALL=C
bash --version | head -n1 | cut -d" " -f2-4
MYSH=$(readlink -f /bin/sh)
echo "/bin/sh -> $MYSH"
echo $MYSH | grep -q bash || echo "ERROR: /bin/sh does not point to bash"
unset MYSH

echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-
bison --version | head -n1

if [ -h /usr/bin/yacc ]; then
  echo "/usr/bin/yacc -> `readlink -f /usr/bin/yacc`";
elif [ -x /usr/bin/yacc ]; then
  echo yacc is `/usr/bin/yacc --version | head -n1`
else
  echo "yacc not found"
fi

echo -n "Coreutils: "; chown --version | head -n1 | cut -d")" -f2
diff --version | head -n1
find --version | head -n1
gawk --version | head -n1

if [ -h /usr/bin/awk ]; then
  echo "/usr/bin/awk -> `readlink -f /usr/bin/awk`";
elif [ -x /usr/bin/awk ]; then
  echo awk is `/usr/bin/awk --version | head -n1`
else
  echo "awk not found"
fi
```

```
gcc --version | head -n1
g++ --version | head -n1
grep --version | head -n1
gzip --version | head -n1
cat /proc/version
m4 --version | head -n1
make --version | head -n1
patch --version | head -n1
echo Perl `perl -V:version`
python3 --version
sed --version | head -n1
tar --version | head -n1
makeinfo --version | head -n1  # texinfo version
xz --version | head -n1

echo 'int main(){}' > dummy.c && g++ -o dummy dummy.c
if [ -x dummy ]
  then echo "g++ compilation OK";
  else echo "g++ compilation failed"; fi
rm -f dummy.c dummy
EOF

bash version-check.sh
```

2.3. Bygge LFS i etapper

LFS er designet for å bygges i én økt. Det er det instruksjonene forutsetter, at systemet ikke vil bli slått av under prosessen. Det betyr ikke at systemet må gjøres i en økt. Problemet er at visse prosedyrer må gjenopprettes etter en omstart hvis LFS gjenopptas på forskjellige punkter.

2.3.1. Kapitler 1–4

Disse kapitlene er utført på vertssystemet. Ved omstart, vær forsiktig med følgende:

- Prosedyrer utført som `root` brukeren etter seksjon 2.4 må ha LFS miljøvariabelen satt *FOR BRUKEREN ROOT*.

2.3.2. Kapitler 5–6

- /mnt/lfs partisjonen må være montert.
- Disse to kapitlene *må* gjøres som bruker `lfs`. En **su - lfs** må gjøres før noen oppgaver i disse kapitlene. Hvis du ikke gjør det, risikerer du å installere pakker til vertssystemet, og potensielt gjøre det ubrukelig.
- Prosedyrene i General Compilation Instructions er kritiske. Hvis det er noen tvil om installerte pakker, sørg for at tidligere utpakkede tarballs fjernes, pakk deretter ut pakkefilene på nytt og fullfør alle instruksjonene i den delen.

2.3.3. Kapitler 7–10

- /mnt/lfs partisjonen må være montert.
- Noen få operasjoner, fra “Skifte eierskap” for “Gå inn i Chroot miljøet” må gjøres som `root` brukeren, med LFS miljøvariabel satt for `root`brukeren.

- Når du går inn i chroot, må LFS miljøvariabelen angis til `root`. LFS variabelen brukes ikke i etterkant.
- De virtuelle filsystemene må være montert. Dette kan gjøres før eller etter at chroot er gått inn i, ved å bytte til en virtuell vertsterminal og som `root`, kjøre kommandoene i Section 7.3.2, “Montering og fylling av /dev” og Section 7.3.3, “Montering av virtuelle kjernefilsystemer”.

2.4. Opprette en ny partisjon

Som de fleste andre operativsystemer er LFS vanligvis installert på en dedikert partisjon. Den anbefalte tilnærmingen til å bygge et LFS system er å bruke en tilgjengelig tom partisjon eller, hvis du har nok upartisjonert plass, å lage en.

Et minimalt system krever en partisjon på rundt 10 gigabyte (GB). Dette er nok til å lagre alle kildetarballene og kompilere pakkene. Men hvis LFS systemet er ment å være det primære Linux systemet, vil tilleggsprogramvare sannsynligvis bli installert som vil kreve ekstra plass. En 30 GB partisjon er en rimelig størrelse for å sørge for nok plass. LFS systemet i seg selv vil ikke ta så mye plass. En stor del av dette kravet er å sørge for tilstrekkelig ledig midlertidig lagring samt for å legge til flere funksjoner etter at LFS er fullført. I tillegg kompilering av pakker kan kreve mye diskplass som vil bli gjenvunnet etter at pakken er installert.

Fordi det ikke alltid er nok minne (RAM) tilgjengelig for kompileringsprosesser er det en god idé å bruke en liten diskpartisjon som `swap` plass. Dette brukes av kjernen for å lagre sjelden brukte data og la mer minne være tilgjengelig for aktive prosesser. `swap` partisjon for et LFS system kan være det samme som det som brukes av vertssystemet, i det tilfellet er det ikke er nødvendig å opprette en annen.

Start et diskpartisjoneringsprogram som f.eks **cfdisk** eller **fdisk** med et kommandolinjealternativ som navngir harddisken som den nye partisjonen vil bli opprettet på—for eksempel `/dev/sda` for den primære diskstasjonen. Lag en innebygd Linux partisjon og en `swap` partisjon, hvis nødvendig. Vennligst referere til `cfdisk(8)` eller `fdisk(8)` hvis du vet ennå ikke vet hvordan du bruker programmene.

Note

For erfarne brukere er andre partisjoneringsordninger mulig. Det nye LFS systemet kan være på et programvare *RAID* matrise eller en *LVM* logisk volum. Noen av disse alternativene krever imidlertid *initramfs*, som er et avansert emne. Disse partisjoneringsmetodene anbefales ikke for førstegangs LFS brukere.

Husk betegnelsen på den nye partisjonen (f.eks., `sda5`). Denne boken vil referere til dette som LFS partisjonen. Husk også betegnelsen på `swap` partisjonen. Disse navnene vil være nødvendig senere for `/etc/fstab` filen.

2.4.1. Andre partisjonsproblemer

Forespørsler om råd om systempartisjonering legges ofte ut på LFS E-post lister. Dette er et høyst subjektivt tema. Standard for de fleste distribusjoner er å bruke hele stasjonen med unntak av en liten partisjon til vekselminne. Dette er ikke optimalt for LFS av flere grunner. Det reduserer fleksibiliteten, gjør deling av data på tvers av flere distribusjoner eller LFS bygg vanskeligere, gjør sikkerhetskopiering mer tidkrevende, og kan kaste bort diskplass gjennom ineffektiv allokering av filsystemstrukturer.

2.4.1.1. Rotpartisjonen

En rot LFS partisjon (ikke å forveksle med `/root` mappen) av tjue gigabyte er et godt kompromiss for de fleste systemer. Det gir nok plass til å bygge LFS og det meste av BLFS, men er liten nok til at flere partisjoner kan enkelt lages for eksperimentering.

2.4.1.2. Vekselminnepartisjonen

De fleste distribusjoner oppretter automatisk et vekselminnepartisjon. Som regel er den anbefalte størrelsen på vekselminnepartisjonen omtrent det dobbelte av fysisk RAM, men dette er sjelden nødvendig. Hvis diskplassen er begrenset, hold vekselminnepartisjonen til to gigabyte og overvåk mengden disk veksling.

Hvis du vil bruke dvalefunksjonen (suspend-to-disk) i Linux, den skriver ut innholdet i RAM til vekselminnepartisjonen før den slår av maskinen. I dette tilfellet bør størrelsen på vekselminnepartisjonen være minst like stor som systemets installerte RAM.

Bruk av vekselminne er aldri bra. For mekaniske harddisker kan du generelt fortelle om et system veksler ved å bare lytte til diskaktivitet og observere hvordan systemet reagerer på kommandoer. For en SSD stasjon vil du ikke kunne høre veksling, men du kan se hvor mye vekslingsplass som brukes ved å bruke **top** eller **free** programmene. Bruken av en SSD stasjon for en vekselminnepartisjon bør unngås hvis mulig. Den første reaksjon på veksling bør være å se etter en urimelig kommando som f.eks prøver å redigere en fil på fem gigabyte. Hvis veksling blir normalt , er den beste løsningen å kjøpe mer RAM til ditt system.

2.4.1.3. Grub Bios partisjonen

Hvis *oppstartsdisken* har blitt partisjonert med en GUID Partisjons Tabell (GPT), da må en liten, vanligvis 1 MB, partisjon bli opprettet hvis den ikke eksisterer allerede. Denne partisjonen er ikke formatert, men må være tilgjengelig for GRUB for å bruke under installasjonen av oppstarts lasteren. Denne partisjonen vil normalt være merket 'BIOS Boot' hvis den opprettes av **fdisk** eller har en kode på *EF02* hvis den opprettes ved hjelp av **gdisk**.

Note

Grub Bios partisjonen må være på stasjonen som BIOS bruker for å starte opp systemet. Dette er ikke nødvendigvis den samme stasjonen der LFS rotpartisjon er lokalisert. Disker på et system kan bruke forskjellig partisjonstabelltyper. Kravet til denne partisjonen avhenger bare på partisjonstabelltypen til oppstartsdisken.

2.4.1.4. Bekvemmelig partisjoner

Det er flere andre partisjoner som ikke er påkrevd, men som bør vurderes når du designer et diskoppsett. Følgende liste er ikke utfyllende, men er ment som en veiledning.

- /boot – Sterkt anbefalt. Bruk denne partisjonen til å lagre kjerner og annen oppstartsinformasjon. For å minimere potensielle oppstarts problemer med større disker, gjør dette til den første fysiske partisjonen på din første diskstasjon. En partisjonsstørrelse på 200 megabyte er ganske tilstrekkelig.
- /home – Sterkt anbefalt. Del hjemme mappen og brukertilpasning på tvers av flere distribusjoner eller LFS bygginger. Størrelsen er vanligvis ganske stor og avhenger av tilgjengelig disk plass.
- /usr – I LFS, `/bin`, `/lib`, og `/sbin` er symbolkoblinger til deres motpart i `/usr`. Så `/usr` inneholder alle binærfiler nødvendig for at systemet skal kjøre. For LFS en egen partisjon for `/usr` er normalt ikke nødvendig. Hvis du uansett trenger det, bør du lage en partisjon som er stor nok til å passe til alle programmer og biblioteker i systemet. Rotpartisjonen kan være veldig liten (kanskje bare én gigabyte) i denne konfigurasjonen, så det er det egnet for en tynn klient eller diskløs arbeidsstasjon (hvor `/usr` monteres fra en fjern server). Du bør imidlertid merke deg at initramfs (ikke dekket av LFS) vil være nødvendig for å starte et system med separat `/usr` partisjon.
- /opt – Denne mappen er mest nyttig for BLFS der flere installasjoner av store pakker som Gnome eller KDE kan installeres uten å bygge inn filene i /usr hierarkiet. Hvis den brukes, er 5 til 10 gigabyte generelt tilstrekkelig.

- /tmp – En separat /tmp mappe er sjelden, men nyttig hvis du konfigurerer en tynn klient. Denne partisjonen, hvis den brukes, vil vanligvis ikke trenge å overstige et par gigabyte.
- /usr/src – Denne partisjonen er veldig nyttig for å gi en plassering for å lagre BLFS kildefiler og dele dem på tvers av LFS bygg. Den kan også brukes som lokasjon for å bygge BLFS pakker. En rimelig stor partisjon på 30-50 gigabyte gir god plass.

Enhver separat partisjon du vil ha automatisk montert ved oppstart må spesifiseres i `/etc/fstab`. Detaljer om hvordan du spesifiserer partisjoner vil bli diskutert i Section 10.2, “Opprette /etc/fstab filen”.

2.5. Opprette et filsystem på partisjonen

Nå som en tom partisjon er satt opp, kan filsystemet bli opprettet. LFS kan bruke et hvilket som helst filsystem som gjenkjennes av Linux kjernen, men de vanligste typene er ext3 og ext4. Valget av filsystem kan være kompleks og avhenger av egenskapene til filene og størrelsen på partisjonen. For eksempel:

ext2
: passer for små partisjoner som oppdateres sjelden slik som /boot.

ext3
: er en oppgradering til ext2 som inkluderer en loggføring for å hjelpe til med å gjenopprette partisjonens status i tilfelle en uren avslutning. Det er ofte brukt som et generelt filsystem.

ext4
: er den nyeste versjonen av ext filsystemfamilien til partisjonstyper. Det gir flere nye funksjoner, inkludert nano-sekunders tidsstempler, opprettelse og bruk av veldig store filer (16 TB), og hastighetsforbedringer.

Andre filsystemer, inkludert FAT32, NTFS, ReiserFS, JFS og XFS er nyttig for spesialiserte formål. Mer informasjon om disse filsystemene finner du på *http://en.wikipedia.org/wiki/Comparison_of_file_systems*.

LFS antar at rotfilsystemet (/) er av typen ext4. Å lage et `ext4` filsystemet på LFS partisjonen, kjør følgende:

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

Erstatt *<xxx>* med navnet på LFS partisjonen.

Hvis du bruker en eksisterende `swap` partisjon, er det ikke nødvendig å formatere den. Hvis en ny `swap` partisjonen ble opprettet, må den initialiseres med denne kommandoen:

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

Erstatt *<yyy>* med navnet på `swap` partisjonen.

2.6. Stille inn $LFS variabelen

Gjennom hele denne boken, miljøvariabelen `LFS` vil brukes flere ganger. Du bør sørge for at denne variabelen alltid er definert gjennom hele LFS byggeprosessen. Det bør settes til navnet på mappen hvor du skal bygge LFS systemet ditt - vi vil bruke `/mnt/lfs` som et eksempel, men mappevalg er opp til deg. Hvis du bygger LFS på en separat partisjon, vil denne mappen være monteringspunktet for partisjonen. Velg en mappeplassering og sett variabelen med følgende kommando:

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

Å ha dette variabelsettet er fordelaktig ved at kommandoer som f.eks **mkdir -v $LFS/tools** kan skrives bokstavelig. Skallet vil automatisk erstatte "$LFS" med "/mnt/lfs" (eller hva variabelen ble satt til) når den behandler kommandolinjen.

Caution

Ikke glem å sjekke at `LFS` er satt når du forlater og går inn i det nåværende arbeidsmiljøet igjen (for eksempel når du gjør en **su** til `root` eller en annen bruker). Sjekk at `LFS` variabelen er satt opp skikkelig med:

```
echo $LFS
```

Sørg for at utdataene viser banen til LFS systemets bygge plassering, som er `/mnt/lfs` hvis gitt eksempel ble fulgt. Hvis utdaten er feil, bruk kommandoen gitt tidligere på denne siden for å sette `$LFS` til det riktige mappenavnet.

Note

En måte å sikre at `LFS` variabelen alltid er satt er å redigere `.bash_profile` fil i både din personlig hjemmemappe og i `/root/.bash_profile` og skriv inn export kommandoen ovenfor. I tillegg, skallet spesifisert i `/etc/passwd` fil for alle brukere som trenger `LFS` variabelen må være bash for å sikre at `/root/.bash_profile` filen er innlemmet som en del av påloggingsprosessen.

En annen vurdering er metoden som brukes for å logge på vertssystemet. Hvis du logger på via en grafisk skjermbehandler, brukerens `.bash_profile` brukes vanligvis ikke når en virtuell terminal startes. I dette tilfellet legger du til export kommandoen til filen `.bashrc` for brukeren og `root`. I tillegg, noen distribusjoner har instruksjoner om å ikke kjøre `.bashrc` instruksjoner i en ikke interaktiv bash påkallelse. Sørg for å legge til export kommandoen før testen for ikke interaktiv bruk.

2.7. Montering av den nye partisjonen

Nå som et filsystem er opprettet, må partisjonen gjøres tilgjengelig. For å gjøre dette, må partisjonen være montert på et valgt monteringspunkt. For formålet med denne boken er det antatt at filsystemet er montert under mappen spesifisert av `LFS` miljøvariabel som beskrevet i forrige avsnitt.

Opprett monteringspunktet og monter LFS filsystemet ved å kjøre:

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

Erstatt *<xxx>* med betegnelsen for LFS partisjon.

Hvis du bruker flere partisjoner for LFS (f.eks. en for `/` og en annen for `/home`), monter dem med:

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

Erstatt *<xxx>* og *<yyy>* med riktige partisjons navn.

Sørg for at de nye partisjonene ikke er montert med tillatelser som er for restriktiv (som f.eks `nosuid` eller `nodev` alternativer). Kjør **mount** kommandoen uten noen parametere for å se hvilke alternativer som er satt for den monterte LFS partisjonen. Hvis `nosuid` og/eller `nodev` er satt, partisjonene må monteres på nytt.

Warning

Instruksjonene ovenfor forutsetter at du ikke starter datamaskinen på nytt din gjennom hele LFS prosessen. Hvis du slår av systemet, enten må du montere LFS partisjonen på nytt hver gang du starter byggeprosessen på nytt eller modifisere vertssystemets /etc/fstab-fil til å automatisk monter den på nytt ved oppstart. For eksempel:

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

Hvis du bruker flere valgfrie partisjoner, sørg for å legge dem til også.

Hvis du bruker en `swap` partisjon, sørg for at den er aktivert, bruk **swapon** kommandoen:

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

Erstatt *<zzz>* med navnet på `swap` partisjonen.

Nå som det er etablert et sted å jobbe, er det på tide å laste ned pakkene.

Chapter 3. Pakker og oppdateringer

3.1. Introduksjon

Dette kapittelet inneholder en liste over pakker som må lastes ned for å bygge et grunnleggende Linux system. De oppførte versjonsnumrene tilsvarer versjoner av programvaren som er kjent for å fungere, og denne boken er basert på deres bruk. Vi anbefaler på det sterkeste å ikke bruke forskjellige versjoner fordi konstruksjonens kommandoer for én versjon kanskje ikke fungerer med en annen versjon, med mindre annen versjon er spesifisert av en LFS errata eller sikkerhetsrådgivning. De nyeste pakkeversjonene kan også ha problemer som krever løsninger. Disse løsningene vil bli utviklet og stabilisert i utviklingsversjon av boken.

For noen pakker, utgivelsens tarball og (Git eller SVN) øyeblikksbilde fra depotets tarball for denne utgivelsen kan publiseres med lignende filnavn. En utgivelses tarball inneholder genererte filer (for eksempel, **configure** skript generert av **autoconf**), i tillegg til innholdet i tilsvarende øyeblikksbilde av depot. Boken bruker utgivelses tarballer når det er mulig. Bruke et øyeblikksbilde av depot i stedet for en utgivelses tarball spesifisert av boken vil forårsake problemer.

Nedlastingsplasseringer er kanskje ikke alltid tilgjengelige. Hvis en nedlastings plasseringen har endret seg siden denne boken ble publisert, Google (*http://www.google.com/*) gir en nyttig søkemotor for de fleste pakkene. Hvis dette søket ikke lykkes, prøv en alternativ måte å laste ned på *https://www.linuxfromscratch.org/lfs/mirrors.html#files*.

Nedlastede pakker og oppdateringer må oppbevares et sted som er praktisk tilgjengelig gjennom hele bygget. En fungerende mappe er også nødvendig for å pakke ut kildene og bygge dem. `$LFS/sources` kan brukes både som et sted å oppbevare tarballene og oppdateringene og som en arbeids mappe. Ved å bruke denne mappen vil de nødvendige elementene være plassert på LFS partisjonen og vil være tilgjengelig under alle stadier av byggeprosessen.

For å opprette denne mappen, utfør følgende kommando, som bruker `root`, før du starter nedlastingens økten:

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

Gjør denne mappen skrivbar og låst (sticky). “Låst” betyr at selv om flere brukere har skrivetillatelse på en mappe, er det bare eieren av en fil som kan slette filen i en låst mappe. Følgende kommando vil aktivere skrive og låste moduser:

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

Det er flere måter å få tak i alle nødvendige pakker og oppdateringer å bygge LFS:

- Filene kan lastes ned individuelt som beskrevet i neste to avsnitt.
- For stabile versjoner av boken, en tarball av alle nødvendige filer kan lastes ned fra et av LFS filspeilene som er oppført på *https://www.linuxfromscratch.org/mirrors.html#files*.
- Filene kan lastes ned ved hjelp av **wget** og en wget-liste som beskrevet nedenfor.

For å laste ned alle pakkene og oppdateringene ved å bruke *wget-liste* som inndata til **wget** kommanden, bruk:

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

Note

`wget-list` filen nevnt ovenfor henter alle pakker for både sysV og systemd versjonene av LFS. Det er totalt fem ekstra små pakker som ikke er nødvendig for den gjeldende boken. `md5sums` filen nevnt nedenfor er spesifikk for gjeldende bok.

I tillegg, fra og med LFS-7.0, er det en egen fil, *md5sums*, som kan brukes til å bekrefte at alle de riktige pakkene er tilgjengelige før du fortsetter. Legg inn denne filen i `$LFS/sources` og kjør:

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

Denne sjekken kan brukes etter å ha hentet de nødvendige filene med en av de metodene oppført ovenfor.

3.2. Alle pakker

Last ned eller på annen måte skaff deg følgende pakker:

• Acl (2.3.1) - 348 KB:
Hjemmeside: *https://savannah.nongnu.org/projects/acl*
Laste ned: *https://download.savannah.gnu.org/releases/acl/acl-2.3.1.tar.xz*
MD5 sum: `95ce715fe09acca7c12d3306d0f076b2`

• Attr (2.5.1) - 456 KB:
Hjemmeside: *https://savannah.nongnu.org/projects/attr*
Laste ned: *https://download.savannah.gnu.org/releases/attr/attr-2.5.1.tar.gz*
MD5 sum: `ac1c5a7a084f0f83b8cace34211f64d8`

• Autoconf (2.71) - 1,263 KB:
Hjemmeside: *https://www.gnu.org/software/autoconf/*
Laste ned: *https://ftp.gnu.org/gnu/autoconf/autoconf-2.71.tar.xz*
MD5 sum: `12cfa1687ffa2606337efe1a64416106`

• Automake (1.16.5) - 1,565 KB:
Hjemmeside: *https://www.gnu.org/software/automake/*
Laste ned: *https://ftp.gnu.org/gnu/automake/automake-1.16.5.tar.xz*
MD5 sum: `4017e96f89fca45ca946f1c5db6be714`
SHA256 sum: `80facc09885a57e6d49d06972c0ae1089c5fa8f4d4c7cfe5baea58e5085f136d`

• Bash (5.1.16) - 10,277 KB:
Hjemmeside: *https://www.gnu.org/software/bash/*
Laste ned: *https://ftp.gnu.org/gnu/bash/bash-5.1.16.tar.gz*
MD5 sum: `c17b20a09fc38d67fb303aeb6c130b4e`

• Bc (5.2.2) - 428 KB:
Hjemmeside: *https://git.yzena.com/gavin/bc*
Laste ned: *https://github.com/gavinhoward/bc/releases/download/5.2.2/bc-5.2.2.tar.xz*
MD5 sum: `632344cdb052af0e06087bd3b0034126`

• Binutils (2.38) - 23,098 KB:
Hjemmeside: *https://www.gnu.org/software/binutils/*
Laste ned: *https://ftp.gnu.org/gnu/binutils/binutils-2.38.tar.xz*
MD5 sum: `6e39cad1bb414add02b5b1169c18fdc5`

• Bison (3.8.2) - 2,752 KB:
Hjemmeside: *https://www.gnu.org/software/bison/*
Laste ned: *https://ftp.gnu.org/gnu/bison/bison-3.8.2.tar.xz*
MD5 sum: `c28f119f405a2304ff0a7ccdcc629713`

• Bzip2 (1.0.8) - 792 KB:
Laste ned: *https://www.sourceware.org/pub/bzip2/bzip2-1.0.8.tar.gz*
MD5 sum: `67e051268d0c475ea773822f7500d0e5`

• Check (0.15.2) - 760 KB:
Hjemmeside: *https://libcheck.github.io/check*
Laste ned: *https://github.com/libcheck/check/releases/download/0.15.2/check-0.15.2.tar.gz*
MD5 sum: `50fcafcecde5a380415b12e9c574e0b2`

• Coreutils (9.0) - 5,482 KB:
Hjemmeside: *https://www.gnu.org/software/coreutils/*
Laste ned: *https://ftp.gnu.org/gnu/coreutils/coreutils-9.0.tar.xz*
MD5 sum: `0d79ae8a6124546e3b94171375e5e5d0`

• D-Bus (1.12.20) - 2,048 KB:
Hjemmeside: *https://www.freedesktop.org/wiki/Software/dbus*
Laste ned: *https://dbus.freedesktop.org/releases/dbus/dbus-1.12.20.tar.gz*
MD5 sum: `dfe8a71f412e0b53be26ed4fbfdc91c4`

• DejaGNU (1.6.3) - 608 KB:
Hjemmeside: *https://www.gnu.org/software/dejagnu/*
Laste ned: *https://ftp.gnu.org/gnu/dejagnu/dejagnu-1.6.3.tar.gz*
MD5 sum: `68c5208c58236eba447d7d6d1326b821`

• Diffutils (3.8) - 1,548 KB:
Hjemmeside: *https://www.gnu.org/software/diffutils/*
Laste ned: *https://ftp.gnu.org/gnu/diffutils/diffutils-3.8.tar.xz*
MD5 sum: `6a6b0fdc72acfe3f2829aab477876fbc`

• E2fsprogs (1.46.5) - 9,307 KB:
Hjemmeside: *http://e2fsprogs.sourceforge.net/*
Laste ned: *https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.46.5/e2fsprogs-1.46.5.tar.gz*
MD5 sum: `3da91854c960ad8a819b48b2a404eb43`

• Elfutils (0.186) - 9,015 KB:
Hjemmeside: *https://sourceware.org/elfutils/*
Laste ned: *https://sourceware.org/ftp/elfutils/0.186/elfutils-0.186.tar.bz2*
MD5 sum: `2c095e31e35d6be7b3718477b6d52702`

• Expat (2.4.6) - 444 KB:
Hjemmeside: *https://libexpat.github.io/*
Laste ned: *https://prdownloads.sourceforge.net/expat/expat-2.4.6.tar.xz*
MD5 sum: `22a30c888752fdda9f8dd1b7281c54b0`

Note

Oppstrøms kan fjerne tarballs av de spesifikke utgivelsene av Expat når disse utgivelsene inneholder en sikkerhetssårbarhet. du bør referere til *LFS sikkerhetsrådgivning* for å finne ut hvilken versjon (med sikkerhetsproblemet løst) som bør bli brukt. Du kan laste ned den sårbare versjonen fra et speil, men det anbefales ikke.

• Expect (5.45.4) - 618 KB:
Hjemmeside: *https://core.tcl.tk/expect/*
Laste ned: *https://prdownloads.sourceforge.net/expect/expect5.45.4.tar.gz*
MD5 sum: `00fce8de158422f5ccd2666512329bd2`

• File (5.41) - 1040 KB:
Hjemmeside: *https://www.darwinsys.com/file/*
Laste ned: *https://astron.com/pub/file/file-5.41.tar.gz*
MD5 sum: `18233bb0a0089dfdc7dfbc93b96f231b`

• Findutils (4.9.0) - 1,999 KB:
Hjemmeside: *https://www.gnu.org/software/findutils/*
Laste ned: *https://ftp.gnu.org/gnu/findutils/findutils-4.9.0.tar.xz*
MD5 sum: `4a4a547e888a944b2f3af31d789a1137`

• Flex (2.6.4) - 1,386 KB:
Hjemmeside: *https://github.com/westes/flex*
Laste ned: *https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz*
MD5 sum: `2882e3179748cc9f9c23ec593d6adc8d`

• Gawk (5.1.1) - 3,075 KB:
Hjemmeside: *https://www.gnu.org/software/gawk/*
Laste ned: *https://ftp.gnu.org/gnu/gawk/gawk-5.1.1.tar.xz*
MD5 sum: `83650aa943ff2fd519b2abedf8506ace`

• GCC (11.2.0) - 78,996 KB:
Hjemmeside: *https://gcc.gnu.org/*
Laste ned: *https://ftp.gnu.org/gnu/gcc/gcc-11.2.0/gcc-11.2.0.tar.xz*
MD5 sum: `31c86f2ced76acac66992eeedce2fce2`
SHA256 sum: `d08edc536b54c372a1010ff6619dd274c0f1603aa49212ba20f7aa2cda36fa8b`

• GDBM (1.23) - 1,092 KB:
Hjemmeside: *https://www.gnu.org/software/gdbm/*
Laste ned: *https://ftp.gnu.org/gnu/gdbm/gdbm-1.23.tar.gz*
MD5 sum: `8551961e36bf8c70b7500d255d3658ec`

• Gettext (0.21) - 9,487 KB:
Hjemmeside: *https://www.gnu.org/software/gettext/*
Laste ned: *https://ftp.gnu.org/gnu/gettext/gettext-0.21.tar.xz*
MD5 sum: `40996bbaf7d1356d3c22e33a8b255b31`

• Glibc (2.35) - 17,741 KB:
Hjemmeside: *https://www.gnu.org/software/libc/*
Laste ned: *https://ftp.gnu.org/gnu/glibc/glibc-2.35.tar.xz*
MD5 sum: `dd571c67d85d89d7f60b854a4e207423`

• GMP (6.2.1) - 1,980 KB:
Hjemmeside: *https://www.gnu.org/software/gmp/*
Laste ned: *https://ftp.gnu.org/gnu/gmp/gmp-6.2.1.tar.xz*
MD5 sum: `0b82665c4a92fd2ade7440c13fcaa42b`

• Gperf (3.1) - 1,188 KB:
Hjemmeside: *https://www.gnu.org/software/gperf/*
Laste ned: *https://ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz*
MD5 sum: `9e251c0a618ad0824b51117d5d9db87e`

• Grep (3.7) - 1,603 KB:
Hjemmeside: *https://www.gnu.org/software/grep/*
Laste ned: *https://ftp.gnu.org/gnu/grep/grep-3.7.tar.xz*
MD5 sum: `7c9cca97fa18670a21e72638c3e1dabf`

• Groff (1.22.4) - 4,044 KB:
Hjemmeside: *https://www.gnu.org/software/groff/*
Laste ned: *https://ftp.gnu.org/gnu/groff/groff-1.22.4.tar.gz*
MD5 sum: `08fb04335e2f5e73f23ea4c3adbf0c5f`

• GRUB (2.06) - 6,428 KB:
Hjemmeside: *https://www.gnu.org/software/grub/*
Laste ned: *https://ftp.gnu.org/gnu/grub/grub-2.06.tar.xz*
MD5 sum: `cf0fd928b1e5479c8108ee52cb114363`

• Gzip (1.11) - 786 KB:
Hjemmeside: *https://www.gnu.org/software/gzip/*
Laste ned: *https://ftp.gnu.org/gnu/gzip/gzip-1.11.tar.xz*
MD5 sum: `d1e93996dba00cab0caa7903cd01d454`

• Iana-Etc (20220207) - 580 KB:
Hjemmeside: *https://www.iana.org/protocols*
Laste ned: *https://github.com/Mic92/iana-etc/releases/download/20220207/iana-etc-20220207.tar.gz*
MD5 sum: `81d865ce7fe4240d5abed48c3ca5fa9f`

• Inetutils (2.2) - 1,494 KB:
Hjemmeside: *https://www.gnu.org/software/inetutils/*
Laste ned: *https://ftp.gnu.org/gnu/inetutils/inetutils-2.2.tar.xz*
MD5 sum: `de8c1b49cbde2b30e481c61c65357ad4`
SHA256 sum: `01b9a4bc73a47e63f6e8a07b76122d9ad2a2e46ebf14870e9c91d660b5647a22`

• Intltool (0.51.0) - 159 KB:
Hjemmeside: *https://freedesktop.org/wiki/Software/intltool*
Laste ned: *https://launchpad.net/intltool/trunk/0.51.0/+download/intltool-0.51.0.tar.gz*
MD5 sum: `12e517cac2b57a0121cda351570f1e63`

• IPRoute2 (5.16.0) - 843 KB:
Hjemmeside: *https://www.kernel.org/pub/linux/utils/net/iproute2/*
Laste ned: *https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-5.16.0.tar.xz*
MD5 sum: `994c1bad2a24aa9d70e89670c5b5dfcb`

• Jinja2 (3.0.3) - 263 KB:
Hjemmeside: *https://jinja.palletsprojects.com/en/3.0.x/*
Laste ned: *https://files.pythonhosted.org/packages/source/J/Jinja2/Jinja2-3.0.3.tar.gz*
MD5 sum: `b76ae2f0647abebc81e7c03f5fb7b00f`

• Kbd (2.4.0) - 1,095 KB:
Hjemmeside: *https://kbd-project.org/*
Laste ned: *https://www.kernel.org/pub/linux/utils/kbd/kbd-2.4.0.tar.xz*
MD5 sum: `3cac5be0096fcf7b32dcbd3c53831380`

• Kmod (29) - 548 KB:
Laste ned: *https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-29.tar.xz*
MD5 sum: `e81e63acd80697d001c8d85c1acb38a0`

• Less (590) - 348 KB:
Hjemmeside: *https://www.greenwoodsoftware.com/less/*
Laste ned: *https://www.greenwoodsoftware.com/less/less-590.tar.gz*
MD5 sum: `f029087448357812fba450091a1172ab`

• Libcap (2.63) - 171 KB:
Hjemmeside: *https://sites.google.com/site/fullycapable/*
Laste ned: *https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.63.tar.xz*
MD5 sum: `18410cec436f827e698ee9ea16ada9b7`

• Libffi (3.4.2) - 1,320 KB:
Hjemmeside: *https://sourceware.org/libffi/*
Laste ned: *https://github.com/libffi/libffi/releases/download/v3.4.2/libffi-3.4.2.tar.gz*
MD5 sum: `294b921e6cf9ab0fbaea4b639f8fdbe8`

• Libpipeline (1.5.5) - 934 KB:
Hjemmeside: *http://libpipeline.nongnu.org/*
Laste ned: *https://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.5.tar.gz*
MD5 sum: `3e725c76bfea1985e87e851ee50c2e29`

• Libtool (2.4.6) - 951 KB:
Hjemmeside: *https://www.gnu.org/software/libtool/*
Laste ned: *https://ftp.gnu.org/gnu/libtool/libtool-2.4.6.tar.xz*
MD5 sum: `1bfb9b923f2c1339b4d2ce1807064aa5`

• Linux (5.16.9) - 124,577 KB:
Hjemmeside: *https://www.kernel.org/*
Laste ned: *https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.16.9.tar.xz*
MD5 sum: `4d6a704bf3e249ef6189b6f17457084b`

Note

Linux kjernen oppdateres relativt ofte, mange ganger pga oppdagelser av sikkerhetssårbarheter. Den siste tilgjengelige stabile kjerne versjonen kan være brukt, med mindre errata siden sier noe annet.

For brukere med begrenset hastighet eller dyr båndbredde som ønsker å oppdatere Linux kjernen, en grunnlinjeversjon av pakken og oppdateringer kan lastes ned separat. Dette kan spare litt tid eller kostnad for en påfølgende nivåoppgradering av oppdateringer i en mindre utgivelse.

• M4 (1.4.19) - 1,617 KB:
Hjemmeside: *https://www.gnu.org/software/m4/*
Laste ned: *https://ftp.gnu.org/gnu/m4/m4-1.4.19.tar.xz*
MD5 sum: `0d90823e1426f1da2fd872df0311298d`

• Make (4.3) - 2,263 KB:
Hjemmeside: *https://www.gnu.org/software/make/*
Laste ned: *https://ftp.gnu.org/gnu/make/make-4.3.tar.gz*
MD5 sum: `fc7a67ea86ace13195b0bce683fd4469`

• Man-DB (2.10.1) - 1,847 KB:
Hjemmeside: *https://www.nongnu.org/man-db/*
Laste ned: *https://download.savannah.gnu.org/releases/man-db/man-db-2.10.1.tar.xz*
MD5 sum: `b03b76a9a00d0d6b2299b823fba4f579`

• Man-pages (5.13) - 1,752 KB:
Hjemmeside: *https://www.kernel.org/doc/man-pages/*
Laste ned: *https://www.kernel.org/pub/linux/docs/man-pages/man-pages-5.13.tar.xz*
MD5 sum: `3ac24e8c6fae26b801cb87ceb63c0a30`

• MarkupSafe (2.0.1) - 20 KB:
Hjemmeside: *https://markupsafe.palletsprojects.com/en/2.0.x/*
Laste ned: *https://files.pythonhosted.org/packages/source/M/MarkupSafe/MarkupSafe-2.0.1.tar.gz*
MD5 sum: `892e0fefa3c488387e5cc0cad2daa523`

• Meson (0.61.1) - 1,963 KB:
Hjemmeside: *https://mesonbuild.com*
Laste ned: *https://github.com/mesonbuild/meson/releases/download/0.61.1/meson-0.61.1.tar.gz*
MD5 sum: `8ed66d5537275df3defffb66d1fb897f`

• MPC (1.2.1) - 820 KB:
Hjemmeside: *http://www.multiprecision.org/*
Laste ned: *https://ftp.gnu.org/gnu/mpc/mpc-1.2.1.tar.gz*
MD5 sum: `9f16c976c25bb0f76b50be749cd7a3a8`

• MPFR (4.1.0) - 1,490 KB:
Hjemmeside: *https://www.mpfr.org/*
Laste ned: *https://www.mpfr.org/mpfr-4.1.0/mpfr-4.1.0.tar.xz*
MD5 sum: `bdd3d5efba9c17da8d83a35ec552baef`

• Ncurses (6.3) - 3,500 KB:
Hjemmeside: *https://www.gnu.org/software/ncurses/*
Laste ned: *https://invisible-mirror.net/archives/ncurses/ncurses-6.3.tar.gz*
MD5 sum: `a2736befde5fee7d2b7eb45eb281cdbe`

• Ninja (1.10.2) - 209 KB:
Hjemmeside: *https://ninja-build.org/*
Laste ned: *https://github.com/ninja-build/ninja/archive/v1.10.2/ninja-1.10.2.tar.gz*
MD5 sum: `639f75bc2e3b19ab893eaf2c810d4eb4`

• OpenSSL (3.0.1) - 14,660 KB:
Hjemmeside: *https://www.openssl.org/*
Laste ned: *https://www.openssl.org/source/openssl-3.0.1.tar.gz*
MD5 sum: `7d07e849d77d276891edd579a8832bb3`

• Patch (2.7.6) - 766 KB:
Hjemmeside: *https://savannah.gnu.org/projects/patch/*
Laste ned: *https://ftp.gnu.org/gnu/patch/patch-2.7.6.tar.xz*
MD5 sum: `78ad9937e4caadcba1526ef1853730d5`

• Perl (5.34.0) - 12,580 KB:
Hjemmeside: *https://www.perl.org/*
Laste ned: *https://www.cpan.org/src/5.0/perl-5.34.0.tar.xz*
MD5 sum: `df7ecb0653440b26dc951ad9dbfab517`

• Pkg-config (0.29.2) - 1,970 KB:
Hjemmeside: *https://www.freedesktop.org/wiki/Software/pkg-config*
Laste ned: *https://pkg-config.freedesktop.org/releases/pkg-config-0.29.2.tar.gz*
MD5 sum: `f6e931e319531b736fadc017f470e68a`

• Procps (3.3.17) - 985 KB:
Hjemmeside: *https://sourceforge.net/projects/procps-ng*
Laste ned: *https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-3.3.17.tar.xz*
MD5 sum: `d60613e88c2f442ebd462b5a75313d56`

• Psmisc (23.4) - 362 KB:
Hjemmeside: *https://gitlab.com/psmisc/psmisc*
Laste ned: *https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.4.tar.xz*
MD5 sum: `8114cd4489b95308efe2509c3a406bbf`

• Python (3.10.2) - 18,341 KB:
Hjemmeside: *https://www.python.org/*
Laste ned: *https://www.python.org/ftp/python/3.10.2/Python-3.10.2.tar.xz*
MD5 sum: `14e8c22458ed7779a1957b26cde01db9`

• Python Documentation (3.10.2) - 7,102 KB:
Laste ned: *https://www.python.org/ftp/python/doc/3.10.2/python-3.10.2-docs-html.tar.bz2*
MD5 sum: `ffa52f0017baf72df9d32dec785fd6ab`

• Readline (8.1.2) - 2,923 KB:
Hjemmeside: *https://tiswww.case.edu/php/chet/readline/rltop.html*
Laste ned: *https://ftp.gnu.org/gnu/readline/readline-8.1.2.tar.gz*
MD5 sum: `12819fa739a78a6172400f399ab34f81`

• Sed (4.8) - 1,317 KB:
Hjemmeside: *https://www.gnu.org/software/sed/*
Laste ned: *https://ftp.gnu.org/gnu/sed/sed-4.8.tar.xz*
MD5 sum: `6d906edfdb3202304059233f51f9a71d`

• Shadow (4.11.1) - 1,618 KB:
Hjemmeside: *https://shadow-maint.github.io/shadow/*
Laste ned: *https://github.com/shadow-maint/shadow/releases/download/v4.11.1/shadow-4.11.1.tar.xz*
MD5 sum: `5a95ec069aa91508167d02fecafaa912`

• Systemd (250) - 10,856 KB:
Hjemmeside: *https://www.freedesktop.org/wiki/Software/systemd/*
Laste ned: *https://github.com/systemd/systemd/archive/v250/systemd-250.tar.gz*
MD5 sum: `8929beb037c587ada4ed201f19756fe2`

• Systemd Man Pages(250) - 596 KB:
Hjemmeside: *https://www.freedesktop.org/wiki/Software/systemd/*
Laste ned: *https://anduin.linuxfromscratch.org/LFS/systemd-man-pages-250.tar.xz*
MD5 sum: `af3aca39abe4e990cb2a9ac63dcdf506`

Note
Linux From Scratch teamet genererer sin egen tarball av mansider som bruker systemd kilden. Dette gjøres for å unngå unødvendige avhengigheter.

• Tar (1.34) - 2,174 KB:
Hjemmeside: *https://www.gnu.org/software/tar/*
Laste ned: *https://ftp.gnu.org/gnu/tar/tar-1.34.tar.xz*
MD5 sum: `9a08d29a9ac4727130b5708347c0f5cf`

• Tcl (8.6.12) - 10,112 KB:
Hjemmeside: *http://tcl.sourceforge.net/*
Laste ned: *https://downloads.sourceforge.net/tcl/tcl8.6.12-src.tar.gz*
MD5 sum: `87ea890821d2221f2ab5157bc5eb885f`

• Tcl Dokumentasjon (8.6.12) - 1,176 KB:
Laste ned: *https://downloads.sourceforge.net/tcl/tcl8.6.12-html.tar.gz*
MD5 sum: `a0d1a5b60bbb68f2f0bd3066a19c527a`

• Texinfo (6.8) - 4,848 KB:
Hjemmeside: *https://www.gnu.org/software/texinfo/*
Laste ned: *https://ftp.gnu.org/gnu/texinfo/texinfo-6.8.tar.xz*
MD5 sum: `a91b404e30561a5df803e6eb3a53be71`

• Tidssonedata (2021e) - 413 KB:
Hjemmeside: *https://www.iana.org/time-zones*
Laste ned: *https://www.iana.org/time-zones/repository/releases/tzdata2021e.tar.gz*
MD5 sum: `4fdfad906ebc85fef30221c10964cce9`

• Util-linux (2.37.4) - 5,971 KB:
Hjemmeside: *https://git.kernel.org/pub/scm/utils/util-linux/util-linux.git/*
Laste ned: *https://www.kernel.org/pub/linux/utils/util-linux/v2.37/util-linux-2.37.4.tar.xz*
MD5 sum: `755919e658c349cad9e1c7c771742d48`

• Vim (8.2.4383) - 15,622 KB:
Hjemmeside: *https://www.vim.org*
Laste ned: *https://anduin.linuxfromscratch.org/LFS/vim-8.2.4383.tar.gz*
MD5 sum: `3168ff48e382a1201bd0cbd0209bd3e0`

Note
Versjonen av vim endres daglig. For å få den nyeste versjonen, gå til *https://github.com/vim/vim/tags*.

• XML::Parser (2.46) - 249 KB:
Hjemmeside: *https://github.com/chorny/XML-Parser*
Laste ned: *https://cpan.metacpan.org/authors/id/T/TO/TODDR/XML-Parser-2.46.tar.gz*
MD5 sum: `80bb18a8e6240fcf7ec2f7b57601c170`

• Xz Utils (5.2.5) - 1,122 KB:
Hjemmeside: *https://tukaani.org/xz*
Laste ned: *https://tukaani.org/xz/xz-5.2.5.tar.xz*
MD5 sum: `aa1621ec7013a19abab52a8aff04fe5b`

• Zlib (1.2.11) - 457 KB:
Hjemmeside: *https://www.zlib.net/*
Laste ned: *https://zlib.net/zlib-1.2.11.tar.xz*
MD5 sum: `85adef240c5f370b308da8c938951a68`

• Zstd (1.5.2) - 1,892 KB:
Hjemmeside: *https://facebook.github.io/zstd/*
Laste ned: *https://github.com/facebook/zstd/releases/download/v1.5.2/zstd-1.5.2.tar.gz*
MD5 sum: `072b10f71f5820c24761a65f31f43e73`

Total størrelse på disse pakkene: ca 458 MB

3.3. Nødvendige oppdateringer

I tillegg til pakkene kreves det også flere oppdateringer. Disse oppdateringene retter eventuelle feil i pakkene som skal være fikset av vedlikeholderen. Oppdateringene gjør også små modifikasjoner som gjør pakkene lettere å jobbe med. Følgende oppdateringene vil være nødvendig for å bygge et LFS-system:

• Binutils LTO Fix Patch - 3.5 KB:
Last ned: *https://www.linuxfromscratch.org/patches/lfs/11.1/binutils-2.38-lto_fix-1.patch*
MD5 sum: `3df11b6123d5bbdb0fc83862a003827a`

• Bzip2 Documentation Patch - 1.6 KB:
Last ned: *https://www.linuxfromscratch.org/patches/lfs/11.1/bzip2-1.0.8-install_docs-1.patch*
MD5 sum: `6a5ac7e89b791aae556de0f745916f7f`

• Coreutils Internationalization Fixes Patch - 166 KB:
Last ned: *https://www.linuxfromscratch.org/patches/lfs/11.1/coreutils-9.0-i18n-1.patch*
MD5 sum: `1eeba2736dfea013509f9975365e4e32`

• Coreutils Chmod Fix Patch - 3.8 KB:
Last ned: *https://www.linuxfromscratch.org/patches/lfs/11.1/coreutils-9.0-chmod_fix-1.patch*
MD5 sum: `4709df88e68279e6ef357aa819ba5b1a`

• Glibc FHS Patch - 2.8 KB:
Last ned: *https://www.linuxfromscratch.org/patches/lfs/11.1/glibc-2.35-fhs-1.patch*
MD5 sum: `9a5997c3452909b1769918c759eff8a2`

• Kbd Backspace/Delete Fix Patch - 12 KB:
Last ned: *https://www.linuxfromscratch.org/patches/lfs/11.1/kbd-2.4.0-backspace-1.patch*
MD5 sum: `f75cca16a38da6caa7d52151f7136895`

• Systemd Upstream Fixes Patch - 20 KB:
Last ned: *https://www.linuxfromscratch.org/patches/lfs/11.1/systemd-250-upstream_fixes-1.patch*
MD5 sum: `71eac6abdad5fba2039dcd011a9ae5b3`

• Zstd Upstream Fixes Patch - 4 KB:
Last ned: *https://www.linuxfromscratch.org/patches/lfs/11.1/zstd-1.5.2-upstream_fixes-1.patch*
MD5 sum: `a7e576e3f87415fdf388392b257cdcf3`

Total størrelse på disse oppdateringene: ca 213.7 KB

I tillegg til de ovennevnte nødvendige oppdateringene, finnes det en rekke valgfrie oppdateringer laget av LFS fellesskapet. Disse valgfrie oppdateringene løser mindre problemer eller aktiverer funksjonalitet som ikke er aktivert som standard. Les gjerne oppdateringsdatabasen som ligger på *https://www.linuxfromscratch.org/patches/downloads/* og anskaff eventuelle tilleggs oppdateringer som passer dine systembehov.

Chapter 4. Siste forberedelser

4.1. Introduksjon

I dette kapittelet vil vi utføre noen tilleggsoppgaver for å forberede byggingen av det midlertidige systemet. Vi vil lage et sett med mapper i `$LFS` for installasjon av midlertidige verktøy, legg til en uprivilegert bruker for å redusere risikoen, og skape et passende byggemiljø for den brukeren. Det vil også bli forklart tidsenheten vi bruker for å måle hvor lang tid LFS pakker tar å bygge, som er "SBU", og gi litt informasjon om pakkenes testpakker.

4.2. Opprette et begrenset mappeoppsett i LFS filsystemet

Den første oppgaven som utføres i LFS partisjonen er å lage et begrenset mappehierarki slik at programmer kompilert i Kapittel 6 (i tillegg til glibc og libstdc++ i Kapittel 5) kan installeres i deres endelige plassering. Dette er nødvendig for at de midlertidige programmene skal overskrives når du bygger dem igjen i Chapter 8.

Opprett det nødvendige mappeoppsettet ved å kjøre følgende som `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
```

Programmer i Kapittel 6 vil bli kompilert med en krysskompilator (mer detaljer i avsnitt Toolchain Technical Notes). For å skille denne krysskompilatoren fra de andre programmene, vil den bli installert i en spesiell mappe. Opprett denne mappen med:

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

4.3. Legge til LFS brukeren

Når du er logget inn som bruker `root`, kan det å gjøre en enkelt feil skade eller ødelegge et system. Derfor, pakkene i de neste to kapitlene er bygget som en uprivilegert bruker. Du kan bruke ditt eget brukernavn, men for å gjøre det enklere å sette opp en ren arbeidsmiljø, opprett en ny bruker kalt `lfs` som medlem av en ny gruppe (også kalt `lfs`) og bruk denne brukeren under installasjonsprosessen. Som `root`, utfør følgende kommandoer for å legge til den nye brukeren:

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

Betydningen av kommandolinjealternativene:

`-s /bin/bash`
Dette gjør **bash** standard skall for brukeren `lfs`.

`-g lfs`
Dette alternativet legger til bruker `lfs` til gruppe `lfs`.

-m
Dette oppretter en hjemmemappe for `lfs`.

-k /dev/null
Denne parameteren forhindrer mulig kopiering av filer fra et skjelett mappe (standard er `/etc/skel`) ved å endre inndataplasseringen til den spesielle nullenheten.

lfs
Dette er det faktiske navnet på den opprettede brukeren.

For å logge inn som `lfs` (i motsetning til å bytte til bruker `lfs` når logget i som `root`, som ikke krever at `lfs` bruker å ha et passord), gi `lfs` et passord:

```
passwd lfs
```

Bevilg `lfs` full tilgang til alle mapper under `$LFS` ved å gjøre `lfs` eier av mappene:

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

Hvis en egen arbeidsmappe ble opprettet som foreslått, gi bruker `lfs` eierskap til denne mappen:

```
chown -v lfs $LFS/sources
```

Note

I noen vertssystemer fullføres ikke følgende kommando riktig og suspenderer påloggingen til lfs brukeren til bakgrunnen. Hvis ledeteksten "lfs:~$" ikke vises umiddelbart, å skrive inn **fg** kommando vil fikse problemet.

Deretter logger du på som bruker `lfs`. Dette kan gjøres via en virtuell konsoll, gjennom en skjermbehandler eller med følgende erstatt/bytt brukerkommando:

```
su - lfs
```

"`-`" instruerer **su** å starte et påloggingsskall i motsetning til et ikke-påloggingsskall. Forskjellen mellom disse to skjelltypene finner du i detalj i `bash(1)` og **info bash**.

4.4. Sette opp miljøet

Sett opp et godt arbeidsmiljø ved å lage to nye oppstartsfiler for **bash** skallet. Mens du er logget inn som bruker `lfs`, utsted følgende kommando å lage en ny `.bash_profile`:

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

Når du er logget på som bruker `lfs`, det første skallet er vanligvis et *login* skall som leser filen `/etc/profile` fra verten (som sannsynligvis inneholder noen innstillinger og miljøvariabler) og deretter `.bash_profile`. **exec env -i.../bin/bash** kommandoen i `.bash_profile` filen erstatter det kjørende skallet med et nytt med et helt tomt miljø, bortsett fra `HOME`, `TERM`, og `PS1` variabler. Dette sikrer at ingen uønskede og potensielt farlige miljøvariabler fra vertssystemet lekker inn i byggemiljøet. Teknikken som brukes her oppnår målet om å sikre et rent miljø.

Den nye instansen av skallet er et *non-login* skall, som ikke leser, og utfører, innholdet i `/etc/profile` eller `.bash_profile` filer, men heller leser og kjører `.bashrc` filen istedet. Opprett `.bashrc` filen nå:

```
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
```

Betydningen av innstillingene i `.bashrc`

`set +h`

set +h kommandoen slår av **bash**'s hashfunksjon. Hashing er vanligvis en nyttig funksjon—**bash** bruker en hashtabell for å huske banen til kjørbare filer for å unngå å søke i `PATH` gang på gang for å finne den samme kjørbare filen. Imidlertid bør de nye verktøyene brukes så snart de er installert. Ved å slå av hashfunksjonen, vil skallet alltid søke `PATH` når et program kjøres. Som sådan vil skallet finne de nylig kompilerte verktøyene i `$LFS/tools/bin` så snart de er det tilgjengelig uten å huske en tidligere versjon av det samme programmet levert av vertsdistroen, i `/usr/bin` eller `/bin`.

`umask 022`

Å sette brukerfiloppretting smasken (umask) til 022 sikrer at nye opprettede filer og mapper bare kan skrives av eieren, men er lesbar og kjørbar av alle (forutsatt at standardmoduser brukes av `open(2)` systemkall, vil nye filer ende opp med tillatelse modus 644 og mapper med modus 755).

`LFS=/mnt/lfs`

`LFS` variabelen skal settes til det valgte monteringens punkt.

`LC_ALL=POSIX`

`LC_ALL` variabel styrer lokaliseringen av visse programmer, slik at meldingene deres følger konvensjonene i et spesifisert land. Innstillingen `LC_ALL` til “POSIX” eller “C” (de to er likeverdige) sikrer at alt fungerer som forventet i chroot miljøet.

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

`LFS_TGT` variabel setter en ikkestandard, men kompatibel maskin beskrivelse for bruk når du bygger vår krysskompiler og linker og når du krysskompiler vår midlertidige verktøykjede. Mer informasjon finnes i Toolchain Technical Notes.

`PATH=/usr/bin`

Mange moderne Linux distribusjoner har slått sammen `/bin` og `/usr/bin`. Når dette er tilfelle, standard `PATH` variabel må settes bare til `/usr/bin/` for Kapittel 6 miljøet. Når dette ikke er tilfelle, legger følgende linje til `/bin` til stien.

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

Hvis `/bin` er ikke et symbolsk lenke, så må den legges til `PATH` variabelen.

PATH=$LFS/tools/bin:$PATH

Ved å putte `$LFS/tools/bin` foran standard `PATH`, krysskompilatoren installert i begynnelsen av Kapittel 5 blir plukket opp av skallet umiddelbart etter installasjonen. Dette, kombinert med å slå av hashing, begrenser risikoen for at kompilatoren fra verten brukes i stedet for krysskompilator.

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

I Kapittel 5 og Kapittel 6, hvis denne variabelen ikke er satt, **configure** skriptet kan forsøke å laste inn konfigurasjonselementer som er spesifikke for enkelte distribusjoner fra `/usr/share/config.site` på vertssystemet. Overstyr det for å forhindre potensiell forurensning fra verten.

export ...

Mens kommandoene ovenfor har satt noen variabler, i rekkefølge for å gjøre dem synlige innenfor eventuelle underskall, eksporterer vi dem.

Important

Flere kommersielle distribusjoner legger til en ikke dokumentert instansiering of `/etc/bash.bashrc` til initialisering av **bash**. Denne filen har potensial til å endre `lfs` brukerens miljø på måter som kan påvirke byggingen av kritiske LFS pakker. For å sikre at `lfs` brukerens miljø er rent, sjekk for nærvær av `/etc/bash.bashrc` og flytt den ut av veienh vis den er tilstede. Som `root` bruker, kjør:

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

Etter at bruken av `lfs` brukeren er ferdig i begynnelsen av Kapittel 7, du kan gjenopprette `/etc/bash.bashrc` (hvis ønsket).

Legg merke til at LFS Bash pakken vi vil bygge i Section 8.34, “Bash-5.1.16” ikke er konfigurert til å laste eller kjøre `/etc/bash.bashrc`, så denne filen er ubrukelig på et fullført LFS system.

Til slutt, å ha miljøet fullt forberedt for å bygge midlertidige verktøy, hente den nettopp opprettede brukerprofilen:

```
source ~/.bash_profile
```

4.5. Om SBU

Mange vil gjerne vite på forhånd hvor lenge det tar å kompilere og installere hver pakke. Fordi Linux Fra Scratch kan bygges på mange forskjellige systemer, er det umulig å gi nøyaktige tidsanslag. Den største pakken (Glibc) vil ta omtrent 20 minutter på de raskeste systemene, men kan ta opptil tre dager på tregere systemer! I stedet for å oppgi faktiske tider, vil Standard byggenhet (SBU) brukes i stedet.

SBU fungerer som følgende. Den første pakken som skal kompileres fra denne boken er binutils i Kapittel 5. Den tiden det tar å kompilere denne pakken er det som vil bli referert til som Standard byggenhet eller SBU. Alle andre kompileringstider vil bli uttrykt relative til denne tiden.

Tenk for eksempel på en pakke hvis kompileringstid er 4,5 SBU. Dette betyr at hvis et system tok 10 minutter å kompilere og installer det første passet med binutils, vil det ta *omtrent* 45 minutter å bygge denne eksempelpakken. Heldigvis er de fleste byggetidene kortere enn den for binutils.

Generelt er ikke SBU helt nøyaktige fordi de er avhengige av mange faktorer, inkludert vertssystemets versjon av GCC. De er gitt her for å gi et estimat på hvor lang tid det kan ta å installere en pakke, men tall kan variere med så mye som dusinvis av minutter i noen tilfeller.

Note

For mange moderne systemer med flere prosessorer (eller kjerner) kan kompileringstiden for en pakke reduseres ved å utføre en "parallell make" ved å enten sette en miljøvariabel eller fortelle **make** programmer hvor mange prosessorer som er tilgjengelige. For eksempel kan en Intel i5-6500 CPU støtte fire samtidige prosesser med:

```
export MAKEFLAGS='-j4'
```

eller bare bygge med:

```
make -j4
```

Når flere prosessorer brukes på denne måten, vil SBU enhetene i boken variere enda mer enn de normalt ville gjort. I noen tilfeller, make trinnet vil rett og slett mislykkes. Å analysere resultatet av byggeprosessen vil også være vanskeligere fordi linjene i forskjellige prosesser vil være sammenflettet. Hvis du får et problem med et byggetrinn, gå tilbake til et enkelt prosessorbygg for å analysere feilmeldingene på riktig måte.

4.6. Om testpakkene

De fleste pakkene gir en testpakke. Å kjøre testpakken for en nybygd pakke er en god idé fordi den kan gi en “tilregnelighets sjekk” som indikerer at alt er kompilert riktig. En testpakke som består sitt sett med kontroller, beviser vanligvis at pakken fungerer slik utvikleren har tenkt. Det gir imidlertid ingen garanti at pakken er helt feilfri.

Noen testpakker er viktigere enn andre. For eksempel, testpakkene for kjerneverktøykjedepakkene—GCC, binutils, og glibc—er av største betydning på grunn av deres sentrale rolle i et riktig fungerende system. Testpakkene for GCC og glibc kan ta veldig lang tid å fullføre, spesielt på tregere maskinvare, men anbefales på det sterkeste.

Note

Å kjører testpakkene i Kapittel 5 og Kapittel 6 er umulig, siden programmene er kompilert med en krysskompilator, som ikke ment å kunne kjøre på byggeverten.

Et vanlig problem med å kjøre testpakkene for binutils og GCC er å gå tom for pseudoterminaler (PTY). Dette kan resultere i et høyt antall feilende prøver. Dette kan skje av flere grunner, men den mest sannsynlig årsaken er at vertssystemet ikke har `devpts` filsystemet satt opp riktig. Dette spørsmålet diskuteres mer detaljert på *https://www.linuxfromscratch.org/lfs/faq.html#no-ptys*.

Noen ganger vil pakkers testpakker mislykkes, men av årsaker som utviklere er klar over og har ansett som ikke kritiske. Se loggene som finnes på *https://www.linuxfromscratch.org/lfs/build-logs/11.1/* for å bekrefte om disse feilene er forventet eller ikke. Denne siden er gyldig for alle tester i denne boken.

Part III. Bygge LFS Kryssverktøykjede og midlertidige verktøy

Viktig foreløpig materiale

Introduksjon

Denne delen er delt inn i tre stadier: først bygge en krysskompilator og tilhørende biblioteker; for det andre, bruke denne kryssverktøykjeden å bygge flere verktøy på en måte som isolerer dem fra vertens distribusjon; for det tredje, gå inn i chroot miljøet, som forbedrer ytterligere vertsisolasjon, og bygge de resterende verktøyene som trengs for å bygge det endelige systemet.

Important

Med denne delen begynner det virkelige arbeidet med å bygge et nytt system. Det krever mye forsiktighet for å sikre at instruksjonene blir fulgt nøyaktig slik boken viser dem. Du bør prøve å forstå hva de gjør, og uansett hvor ivrig du er etter å fullføre bygget, bør du avstå fra å skrive dem blindt som vist, men les heller dokumentasjon når det er noe du ikke forstår. Hold også styr på skrivingen din og utdata av kommandoer, ved å sende dem til en fil ved å bruke **tee** verktøyet. Dette gir bedre diagnose hvis noe går galt.

Den neste delen gir en teknisk introduksjon til byggeprosessen, mens den følgende inneholder **veldig viktige** generelle instruksjoner.

Verktøykjedens tekniske merknader

Denne delen forklarer noen av begrunnelsen og de tekniske detaljene bak den overordnede byggemetoden. Det er ikke nødvendig å umiddelbart forstå alt i denne delen. Det meste av denne informasjonen vil være klarere etter å ha utført en faktisk konstruksjon. Denne delen kan refereres til når som helst under prosessen.

Det overordnede målet for Kapittel 5 og Kapittel 6 er å produsere et midlertidig område som inneholder et kjent sett med verktøy som kan isoleres fra vertssystemet. Ved bruk av **chroot**, kommandoene i de resterende kapitlene vil være inne i det miljøet, og sikre en ren, problemfri bygging av det nye LFS systemet. Byggeprosessen er designet for å minimere risikoen for nye lesere og gi den mest pedagogiske verdien samtidig.

Byggeprosessen baserer seg på prosessen med *krysskompilering*. Krysskompilering brukes normalt for å bygge en kompilator og dens verktøykjede for en annen maskin enn den som brukes til byggingen. Dette er strengt tatt ikke nødvendig for LFS, siden maskinen der det nye systemet skal kjøre er den samme som den brukt til byggingen. Men krysskompilering har den store fordelen at alt som er krysskompilert ikke avhenger av vertsmiljøet.

Om Krysskompilering

Note

LFS boken er ikke, og inneholder ikke en generell veiledning til å bygge en kryss (eller lokal) verktøykjede. Ikke bruk kommandoen i boken for en kryssverktøykjede som skal brukes til andre formål enn å bygge LFS, med mindre du virkelig forstår hva du gjør.

Krysskompilering involverer noen begreper som fortjener en seksjon for seg selv. Selv om denne delen kan utelates i en første lesning, å komme tilbake til det senere vil være gunstig for din fulle forståelse av prosessen.

La oss først definere noen begreper som brukes i denne sammenhengen:

bygg
: er maskinen der vi bygger programmer. Merk at denne maskinen refereres til som “verten” i andre seksjoner.

vert
: er maskinen/systemet der de bygde programmene skal kjøres. Merk at denne bruken av “host” ikke er den samme som i andre seksjoner.

mål
: brukes kun for kompilatorer. Det er maskinen kompilatoren produserer kode for. Det kan være forskjellig fra både build og host.

Som et eksempel, la oss forestille oss følgende scenario (noen ganger referert til som “Canadian Cross”): vi kan ha en kompilator bare på en treg maskin, la oss kalle det maskin A, og kompilatoren ccA. Vi kan også ha en rask maskin (B), men uten kompilator, og vi ønsker å produsere kode for en annen treg maskin (C). Å bygge en kompilator for maskin C, ville vi ha tre trinn:

Stadie	Bygg	Vert	Mål	Handling
1	A	A	B	bygg krysskompilator cc1 med ccA på maskin A
2	A	B	C	bygg krysskompilator cc2 med cc1 på maskin A
3	B	C	C	bygg kompilator ccC med cc2 på maskin B

Deretter kan alle de andre programmene som trengs av maskin C kompileres ved å bruke cc2 på den raske maskinen B. Merk at med mindre B kan kjøre programmer produsert for C, er det ingen måte å teste de bygde programmene før maskinen C selv kjører. For eksempel, for å teste ccC, vil vi kanskje legge til en fjerde trinn:

Stadie	Bygg	Vert	Mål	Handling
4	C	C	C	bygge om og teste ccC ved å bruke seg selv på maskin C

I eksemplet ovenfor er bare cc1 og cc2 krysskompilatorer, det vil si de produserer kode for en annen maskin enn de de kjører på. De andre kompilatorene ccA og ccC produserer kode for maskinen de kjører på. Slike kompilatorer kalles *lokale* kompilatorer.

Implementering av Krysskompilering for LFS

Note

Nesten alle byggesystemer bruker navn i formen cpu-vendor-kernel-os referert til som maskintripletten. En klok leseren kan lure på hvorfor en “triplett” refererer til et firekomponents navn. Årsaken er historie: i utgangspunktet var tre komponentnavn nok å angi en maskin entydig, men nye maskiner og systemer dukket opp, som viste seg utilstrekkelig. Ordet “triplett” hang igjen. En enkel måte å finne din maskintriplett på er å kjøre **config.guess** skript som følger med kilden for mange pakker. Pakk ut binutils kilder og kjør skriptet: **`./config.guess`** og merk deg utdataen. For eksempel, for en 32-bits Intel-prosessor utdataen vil være *i686-pc-linux-gnu*. På et 64-bit system blir det *x86_64-pc-linux-gnu*.

Vær også oppmerksom på navnet på plattformens dynamiske lenker, ofte referert til som den dynamiske lasteren (ikke å forveksle med standard lenker **ld** som er en del av binutils). Den dynamiske lenkeren levert av Glibc finner og laster de delte bibliotekene som trengs av et program, forbereder programmet for kjøring, og deretter kjører det. Navnet på dynamisk lenker for en 32-bits Intel-maskin er `ld-linux.so.2` og er `ld-linux-x86-64.so.2` for 64-bits systemer. En sikker måte å bestemme navnet på den dynamiske lenkeren på er å inspisere en tilfeldig binær fra vertssystemet ved å kjøre: **`readelf -l <name of binary> | grep interpreter`** og legger merke til utdataen. Den autoritative referansen som dekker alle plattformer er i `shlib-versions` filen i roten til Glibc kildetreet.

For å forfalske en krysskompilering i LFS, navnet på vertstripletten justeres litt ved å endre "vendor" feltet i `LFS_TGT` variabelen. Vi bruker også *`--with-sysroot`* alternativet når du bygger krysslenkeren og krysskompilatoren for å fortelle dem hvor de skal finne de nødvendige vertsfilene. Dette sikrer at ingen av de andre programmene bygget i Kapittel 6 kan lenke til biblioteker på byggemaskinen. Kun to trinn er obligatoriske, og ett til for tester:

Stadie	Bygg	Vert	Mål	Handling
1	pc	pc	lfs	bygg krysskompilator cc1 ved å bruke cc-pc på pc
2	pc	lfs	lfs	bygg kompilator cc-lfs ved å bruke cc1 på pc
3	lfs	lfs	lfs	bygge om og teste cc-lfs ved å bruke seg selv på lfs

I tabellen ovenfor, “på pc” betyr at kommandoene kjøres på en maskin som bruker den allerede installerte distribusjonen. “på lfs” betyr at kommandoene kjøres i et chroot-miljø.

Nå er det mer om krysskompilering: C-språket er ikke bare en kompilator, men definerer også et standardbibliotek. I denne boken blir GNU C-biblioteket, kalt glibc, brukt. Dette biblioteket må kompileres for lfs-maskinen, det vil si ved å bruke krysskompilatoren cc1. Men kompilatoren selv bruker interne bibliotekimplementeringskomplekse instruksjoner som ikke tilgjengelige i assembleranvisningssettet. Dette interne biblioteket heter libgcc, og må være koblet til glibc biblioteket for å være fullt funksjonelt! Videre må standardbiblioteket for C++ (libstdc++) også kobles til glibc. Løsningen på dette kylling og egg problemet er først å bygge en degradert cc1 basert libgcc, som mangler noen funksjoner som tråder og unntakshåndtering, bygge glibc ved å bruke denne degraderte kompilatoren (glibc selv er ikke degradert), og bygg deretter libstdc++. Men dette siste biblioteket vil mangle samme funksjonalitet som libgcc.

Dette er ikke slutten på historien: konklusjonen av det foregående avsnittet er at cc1 ikke er i stand til å bygge en fullt funksjonell libstdc++, men dette er den eneste kompilatoren som er tilgjengelig for å bygge C/C++ bibliotekene under trinn 2! Selvfølgelig, kompilatoren bygget under trinn 2, cc-lfs, ville være i stand til å bygge disse bibliotekene, men (1) byggesystemet til GCC vet ikke at det er brukbart på pc, og (2) å bruke det på pc vil være en fare for å koble til pc-bibliotekene, siden cc-lfs er en lokal kompilator. Så vi må bygge libstdc++ senere, i chroot.

Andre prosedyredetaljer

Krysskompilatoren vil bli installert i en separat `$LFS/tools` mappe, siden det ikke vil være en del av det endelige systemet.

Binutils installeres først fordi **configure** kjøringer av både GCC og Glibc utfører forskjellige funksjonstester på assembleren og lenker for å bestemme hvilke programvarefunksjoner som skal aktiveres eller deaktiveres. Dette er viktigere enn man kanskje først er klar over. En feilkonfigurert GCC eller Glibc kan resultere i en subtilt ødelagt verktøykjede, hvor virkningen av et slikt brudd ikke vises før mot slutten av konstruksjonen av hele distribusjonen. En feil i testserien vil vanligvis fremheve denne feilen før det utføres for mye tilleggsarbeid.

Binutils installerer sin assembler og lenker på to steder, `$LFS/tools/bin` og `$LFS/tools/$LFS_TGT/bin`. Verktøyene i en plassering er hardlenket til den andre. En viktig fasett av lenkeren er bibliotekets søkerekkefølge. Detaljert informasjon kan fås fra **ld** ved å gi den *`--verbose`* flagget. For eksempel, **$LFS_TGT-ld --verbose | grep SEARCH** vil illustrere gjeldende søkestier og rekkefølgen deres. Det viser hvilken filer som er lenket av **ld** ved å kompilere et dummyprogram og gi *`--verbose`* parameteren til lenkeren. For eksempel, **$LFS_TGT-gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded** vil vise alle filene som ble åpnet under koblingen.

Den neste pakken som er installert er GCC. Et eksempel på hva som kan bli sett under kjøringen av **configure** er:

```
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
```

Dette er viktig av grunnene nevnt ovenfor. Det viser også at GCCs konfigureringsskript ikke søker i PATH mapper for å finne hvilke verktøy det skal bruke. Imidlertid under selve kjøringen av **gcc**, er det ikke de samme søkestiene nødvendigvis brukt. For å finne ut hvilke standardlenker **gcc** vil bruke, kjør: **$LFS_TGT-gcc -print-prog-name=ld**.

Detaljert informasjon kan fås fra **gcc** med å gi alternativet *`-v`* på kommandolinjen under kompilering av et dummy-program. For eksempel, **gcc -v dummy.c** Vil vise detaljert informasjon om forprosessorenkompileringen og sammenstillings stadier, inkludert **gcc** sine inkluderte søkestier og deres rekkefølge.

Neste installert er desinfiserte Linux API deklarasjoner (headers). Disse tillater standard C-bibliotek (Glibc) å bruke funksjoner som Linux kjernen vil gi.

Den neste pakken som blir installert er Glibc. Det viktigste hensyn for å bygge Glibc er kompilatoren, binære verktøy og kjernedeklarasjoner. Kompilatoren er generelt ikke et problem siden Glibc vil alltid bruke kompilatoren som er relatert til *--host* parameteret sendt til konfigureringsskriptet; f.eks. i vårt tilfelle kompilatoren vil være **$LFS_TGT-gcc**. De binære verktøyene og kjerne deklarasjoner kan være litt mer kompliserte. Derfor tar vi ingen risiko og bruker de tilgjengelige konfigurasjonsbryterne for å fremtvinge de riktige valgene. Etter kjøring av **configure**, sjekk innholdet i `config.make` filen i `build` mappen for alle viktige detaljer. Legg merke til bruken av *CC="$LFS_TGT-gcc"* (med `$LFS_TGT` utvidet) for å kontrollere hvilke binære verktøy som brukes og bruken av *-nostdinc* og *-isystem* flaggfor å kontrollere kompilatorens inkluderte søkeveier. Disse elementene fremhever et viktig aspekt ved Glibc pakken—den er veldig selvforsynt med tanke på byggemaskineriet og er generelt ikke avhengig av standardinnstillinger for verktøykjeder.

Som nevnt ovenfor, blir standard C++-biblioteket kompilert som neste, etterfulgt i Kapittel 6 av alle programmene som må bygges selv. Installasjonstrinnet for alle disse pakkene bruker `DESTDIR` variabelen for å få programmene til å lande i LFS filsystemet.

Ved slutten av Kapittel 6 den lokale lfs kompilatoren er installert. Første binutils-pass2 blir bygget, med det samme `DESTDIR` installasjon som de andre programmene, deretter konstrueres den andre passeringen av GCC, og utelater libstdc++ og andre ikke-viktige biblioteker. På grunn av en merkelig logikk i GCC konfigureringsskript, `CC_FOR_TARGET` ender opp som **cc** når verten er den samme som målet, men er forskjellig fra byggesystemet. Det er derfor *CC_FOR_TARGET=$LFS_TGT-gcc* er satt eksplisitt inn i konfigureringsalternativene.

Når du kommer inn i chroot-miljøet i Kapittel 7, den første oppgaven er å installere libstdc++. Deretter utføres midlertidige installasjoner av programmer som trengs for riktig betjening av verktøykjeden. Fra dette tidspunktet og fremover er kjerneverktøykjeden selvstendig og selvbetjent. I Chapter 8, bygges, testes og installeres de endelige versjonene av alle pakker som trengs for et fullt funksjonelt system.

Generelle kompileringsinstruksjoner

Ved bygging av pakker er det flere forutsetninger som er gjort innenfor instruksjonene:

- Flere av pakkene oppdateres før kompilering, men kun når oppdateringen er nødvendig for å omgå et problem. En oppdatering er ofte nødvendig i både dette og de følgende kapitlene, men noen ganger på bare ett sted. Vær derfor ikke bekymret hvis instruksjoner for en nedlastet oppdatering vises å være savnet. Advarselsmeldinger om *offset* eller *fuzz* kan også oppstå ved en oppdatering. Ikke bekymre deg for disse advarslene, siden oppdateringen fortsatt var vellykket anvendt.
- Under kompileringen av de fleste pakkene vil det være flere advarsler som ruller forbi på skjermen. Disse er normale og kan trygt bli ignorert. Disse advarslene er slik de vises—advarsler om utdatert, men ikke ugyldig, bruk av C- eller C++-syntaksen. C-standardene endres ganske ofte, og noen pakker bruker fortsatt den eldre standarden. Dette er ikke et problem, men gir en advarsel.
- Sjekk en siste gang at `LFS` miljøvariabelen er riktig satt opp:

```
echo $LFS
```

  Sørg for at utdataen viser banen til LFS partisjonens monterings punkt, som er `/mnt/lfs`, ved bruken av vårt eksempel.

- Til slutt må to viktige punkter understrekes:

Important

Byggeinstruksjonene forutsetter at Systemkrav for vert, inkludert symbolske lenker, har blitt riktig innstilt:

- **bash** er skallet i bruk.
- **sh** er en symbolsk lenke til **bash**.
- **/usr/bin/awk** er en symbolsk lenke til **gawk**.
- **/usr/bin/yacc** er en symbolsk lenke til **bison** eller et lite skript som starter bison.

Important

Å understreke byggeprosessen på nytt:

1. Plasser alle kildene og oppdateringene i en mappe som vil være tilgjengelig fra chroot-miljøet som f.eks `/mnt/lfs/sources/`.
2. Bytt til kildemappen.
3. For hver pakke:
   a. Bruk **tar** programmet, til å pakke ut pakken som skal bygges. I Kapittel 5 og Kapittel 6, sikre at du er *lfs* brukeren når du pakker ut pakken.

      Alle metoder for å få bygget kildekodetreet på sin plass, bortsett fra å trekke ut pakkens tarball, er ikke støttet. Spesielt ved å bruke **cp -R** for å kopiere kildekodetre til et annet sted kan ødelegge koblinger og tidsstempler i kildetreet og forårsake at byggingen feiler.
   b. Bytt til mappen som ble opprettet da pakken ble pakket ut.
   c. Følg bokens instruksjoner for å bygge pakken.
   d. Bytt tilbake til kildemappen.
   e. Slett den utpakkede kildemappen med mindre du blir bedt om noe annet.

Chapter 5. Kompilere en kryssverktøykjede

5.1. Introduksjon

Dette kapittelet viser hvordan du bygger en krysskompilator og dens tilhørende verktøy. Selv om krysskompilering her er forfalsket, er prinsippene det det samme som for en ekte kryssverktøykjede.

Programmene som er kompilert i dette kapittelet vil bli installert under `$LFS/tools` mappe for å beholde dem atskilt fra filene som er installert i de følgende kapitlene. Bibliotekene, på den annen side, er installert på sin endelige plass, siden de er knyttet til systemet vi ønsker å bygge.

5.2. Binutils-2.38 - Pass 1

Binutils pakken inneholder en linker, en assembler og annet verktøy for håndtering av objektfiler.

Omtrentlig byggetid: 1 SBU
Nødvendig diskplass: 620 MB

5.2.1. Installasjon av Kryss Binutils

Note

Gå tilbake og les notatene i avsnittet med tittelen General Compilation Instructions på nytt. Å forstå notatene merket som viktig kan spare deg for mange problemer senere.

Det er viktig at Binutils er den første pakken som blir satt sammen fordi både Glibc og GCC utfører ulike tester på tilgjengelige linker og assembler for å bestemme hvilke av deres egne funksjoner som skal aktiveres.

Binutils dokumentasjonen anbefaler å bygge Binutils i en dedikert byggemappe:

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

Note

For at SBU verdiene som er oppført i resten av boken skal kunne brukes, måler du tiden det tar å bygge denne pakken fra konfigurasjonen, til og med den første installasjonen. For å oppnå dette enkelt, pakk kommandoene inn i en **time** kommando som dette: **`time { ../configure ... && make && make install; }`**.

Forbered nå Binutils til kompilering:

```
../configure --prefix=$LFS/tools \
             --with-sysroot=$LFS \
             --target=$LFS_TGT   \
             --disable-nls       \
             --disable-werror
```

Betydningen av konfigurasjonsalternativene:

`--prefix=$LFS/tools`
Dette forteller konfigurasjonsskriptet å forberede for å installere Binutils programmene i `$LFS/tools` mappen.

`--with-sysroot=$LFS`
For krysskompilering, dette forteller byggesystemet å søke i $LFS etter målsystembibliotekene etter behov.

`--target=$LFS_TGT`
Fordi maskinbeskrivelsen i variabelen `LFS_TGT` er litt annerledes enn verdien som returneres av **config.guess** skriptet, vil denne bryteren fortelle skriptet **configure** om å justere Binutils byggesystem for å bygge en tverrlinker.

`--disable-nls`
Dette deaktiverer internasjonalisering ettersom i18n ikke er nødvendig for de midlertidige verktøyene.

`--disable-werror`
Dette forhindrer byggingen i å stoppe i tilfelle det er advarsler fra vertens kompilator.

Fortsett med å kompilere pakken:

```
make
```

Installer pakken:

```
make install
```

Detaljer om denne pakken finner du i avsnitt Section 8.18.2, “Innhold i Binutils.”

5.3. GCC-11.2.0 - Pass 1

GCC pakken inneholder GNU kompilatorsamlingen, som inkluderer C og C++ kompilatorene.

Omtrentlig byggetid: 11 SBU
Nødvendig diskplass: 3.3 GB

5.3.1. Installasjon av Kryss GCC

GCC krever GMP-, MPFR- og MPC pakkene. Siden disse pakkene kanskje ikke er inkludert i vertsdistribusjonen din, blir de bygget med GCC. Pakk ut hver pakke i GCC-kildemappen, og gi nytt navn til de resulterende mappene slik at GCC byggprosedyrene automatisk bruker dem:

Note

Det er hyppige misforståelser om dette kapittelet. Prosedyrene er de samme som alle andre kapitler som forklart tidligere (Package build instructions). Pakk først ut gcc-tarballen fra kildemappen og bytt deretter til den opprettede mappen. Først da bør du fortsette med instruksjonene nedenfor.

```
tar -xf ../mpfr-4.1.0.tar.xz
mv -v mpfr-4.1.0 mpfr
tar -xf ../gmp-6.2.1.tar.xz
mv -v gmp-6.2.1 gmp
tar -xf ../mpc-1.2.1.tar.gz
mv -v mpc-1.2.1 mpc
```

På x86_64-verter, sett standard mappenavn for 64-bits biblioteker til “lib”:

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

GCC dokumentasjonen anbefaler å bygge GCC i en dedikert byggemappe:

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

Forbered GCC for kompilering:

```
../configure                  \
    --target=$LFS_TGT         \
    --prefix=$LFS/tools       \
    --with-glibc-version=2.35 \
    --with-sysroot=$LFS       \
    --with-newlib             \
    --without-headers         \
    --enable-initfini-array   \
    --disable-nls             \
    --disable-shared          \
    --disable-multilib        \
    --disable-decimal-float   \
    --disable-threads         \
    --disable-libatomic       \
    --disable-libgomp         \
    --disable-libquadmath     \
    --disable-libssp          \
    --disable-libvtv          \
    --disable-libstdcxx       \
    --enable-languages=c,c++
```

Betydningen av konfigurasjonsalternativene:

`--with-glibc-version=2.35`

Dette alternativet spesifiserer versjonen av glibc som vil være brukt på målet. Det er ikke relevant for vertens libc distribusjon fordi alt kompilert av pass1 gcc vil kjøre i chroot miljøet, som er isolert fra libc til vertens distribusjon.

`--with-newlib`

Siden et fungerende C-bibliotek ikke er tilgjengelig ennå, sikrer dette at inhibit_libc-konstanten blir definert når du bygger libgcc. Dette forhindrer kompilering av kode som krever libc støtte.

`--without-headers`

Når du oppretter en komplett tverrkompilator, krever GCC standarddeklarasjoner som er kompatible med målsystemet. For våre formål vil disse deklarasjonene ikke være nødvendige. Denne bryteren hindrer GCC i å lete etter dem.

`--enable-initfini-array`

Denne bryteren tvinger bruk av noen interne datastrukturer som er nødvendige, men som ikke kan oppdages når du bygger en krysskompilator.

`--disable-shared`

Denne bryteren tvinger GCC til å koble sine interne biblioteker statisk. Vi trenger dette fordi de delte bibliotekene krever glibc, som ennå ikke er installert på målsystemet.

`--disable-multilib`

På x86_64 støtter LFS ennå ikke en multilib-konfigurasjon. Denne bryteren er ufarlig for x86.

`--disable-decimal-float, --disable-threads, --disable-libatomic, --disable-libgomp, --disable-libquadmath, --disable-libssp, --disable-libvtv, --disable-libstdcxx`

Disse bryterne deaktiverer støtte for flytende desimal punkter, tråd, libatomic, libgomp, libquadmath, libssp, henholdsvis libvtv og C++ standardbiblioteket. Disse funksjonene klarer ikke å kompilere når du bygger en tverrkompilator og er ikke nødvendig for oppgaven med å krysskompilere den midlertidige libc.

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

Dette alternativet sikrer at bare C- og C ++ -kompilatorene blir bygget. Dette er de eneste språkene som trengs nå.

Kompiler GCC ved å kjøre:

```
make
```

Installer pakken:

```
make install
```

Denne versjonen av GCC har installert et par interne systemoverskrifter. Normalt vil en av dem, `limits.h`, i sin tur inkludere den tilsvarende system `limits.h` systemoverskrift, i dette tilfellet, `$LFS/usr/include/limits.h`. På tidspunktet for denne byggingen av GCC eksisterer imidlertid ikke `$LFS/usr/include/limits.h` så den interne overskriften som nettopp har blitt installert er en delvis, selvstendig fil og inkluderer ikke de utvidede funksjonene til systemoverskriften. Dette er tilstrekkelig for å bygge glibc, men den fullstendige interne overskriften vil være nødvendig senere. Lag en fullversjon av den interne overskriften ved å bruke en kommando som er identisk med det GCC-byggesystemet gjør under normale omstendigheter:

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

Detaljer om denne pakken finner du i avsnitt Section 8.26.2, “Innhold i GCC.”

5.4. Linux-5.16.9 API Headers

Linux API deklarasjonene (i linux-5.16.9.tar.xz) eksponerer kjernens API for bruk av Glibc.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 1.2 GB

5.4.1. Installasjon av Linux API deklarasjoner

Linux-kjernen må eksponere et applikasjonsprogrammeringsgrensesnitt (API) som systemets C bibliotek (Glibc i LFS) kan bruke. Dette har blitt gjort ved å rense ulike C deklarasjonsfiler som er i Linux kjernekilde tarball.

Sørg for at det ikke er noen gamle filer innebygd i pakken:

```
make mrproper
```

Trekk nå ut de brukersynlige kjernedeklarasjonene fra kilden. Det anbefalte målet “headers_install” Kan ikke brukes, fordi det krever rsync, som kanskje ikke er tilgjengelig. Overskriftene plasseres først i `./usr`, deretter kopiert til nødvendig plassering.

```
make headers
find usr/include -name '.*' -delete
rm usr/include/Makefile
cp -rv usr/include $LFS/usr
```

5.4.2. Innhold i Linux API deklarasjoner

Installerte overskrifter: /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, og /usr/include/xen/*.h
Installerte mapper: /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, og /usr/include/xen

Korte beskrivelser

`/usr/include/asm/*.h`	Linux API ASM deklarasjoner
`/usr/include/asm-generic/*.h`	Linux API ASM Generiske deklarasjoner
`/usr/include/drm/*.h`	Linux API DRM deklarasjoner
`/usr/include/linux/*.h`	Linux API Linux deklarasjoner
`/usr/include/misc/*.h`	Linux API Diverse deklarasjoner
`/usr/include/mtd/*.h`	Linux API MTD deklarasjoner
`/usr/include/rdma/*.h`	Linux API RDMA deklarasjoner
`/usr/include/scsi/*.h`	Linux API SCSI deklarasjoner
`/usr/include/sound/*.h`	Linux API Lyd deklarasjoner
`/usr/include/video/*.h`	Linux API Video deklarasjoner
`/usr/include/xen/*.h`	Linux API Xen deklarasjoner

5.5. Glibc-2.35

Glibc pakken inneholder C hovedbiblioteket. Dette biblioteket tilbyr de grunnleggende rutinene for tildeling av minne, søk i kataloger, åpne og lukke filer, lese og skrive filer, strenghåndtering, mønstertilpasning, aritmetikk og så videre.

Omtrentlig byggetid: 4.3 SBU
Nødvendig diskplass: 818 MB

5.5.1. Installasjon av Glibc

Først oppretter du en symbolsk lenke for LSB kompalitet. I tillegg, for x86_64 oppretter du en symbolsk kompatibilitetskobling som kreves for korrekt operasjon av den dynamiske biblioteklasteren:

```
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

Kommandoen ovenfor er riktig. **ln** kommandoen har noen få syntaktiske versjoner, så sørg for å sjekke **info coreutils ln** og `ln(1)` før du rapporterer det du kanskje tror er en feil.

Noen av Glibc-programmene bruker FHS inkompatible `/var/db` mappen for å lagre deres kjøretidsdata. Bruk følgende oppdatering for å få slike programmer til å lagre sine kjøretidsdata på de FHS kompatible stedene:

```
patch -Np1 -i ../glibc-2.35-fhs-1.patch
```

Glibc dokumentasjonen anbefaler å bygge Glibc i en dedikert byggemappe:

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

Sørg for at **ldconfig** og **sln** verktøy er installert i `/usr/sbin`:

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

Deretter forbereder du Glibc for kompilering:

```
../configure                             \
      --prefix=/usr                      \
      --host=$LFS_TGT                    \
      --build=$(../scripts/config.guess) \
      --enable-kernel=3.2                \
      --with-headers=$LFS/usr/include    \
      libc_cv_slibdir=/usr/lib
```

Betydningen av konfigureringsalternativene:

`--host=$LFS_TGT, --build=$(../scripts/config.guess)`
Den kombinerte effekten av disse bryterne er at Glibcs byggesystem konfigurerer seg selv til å være krysskompilert, ved hjelp av krysskoblingen og krysskompilator i `$LFS/tools`.

`--enable-kernel=3.2`
Dette forteller Glibc å kompilere biblioteket med støtte til 3.2 og senere Linux kjerner. Løsninger for eldre kjerner er ikke aktivert.

`--with-headers=$LFS/usr/include`
Dette forteller Glibc å kompilere seg selv mot deklarasjonene nylig installert i mappen $LFS/usr/include, slik at den vet nøyaktig hvilke funksjoner kjernen har og kan optimalisere seg selv deretter.

`libc_cv_slibdir=/usr/lib`
Dette sikrer at biblioteket er installert i /usr/lib i stedet for standard /lib64 på 64-bits maskiner.

I løpet av dette stadiet kan følgende advarsel vises:

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

Den manglende eller inkompatible **msgfmt** programmet er generelt ufarlig. Dette **msgfmt** programmet er en del av Gettext pakken som vertsdistribusjonen skal gi.

Note

Det har vært rapporter om at denne pakken kan mislykkes når bygning som et "parallell make". Hvis dette skjer, kjør make kommandoen på nytt med et "-j1"-alternativ.

Kompiler pakken:

```
make
```

Installer pakken:

Warning

Hvis `LFS` ikke er riktig innstilt, og til tross for anbefalinger, bygger du som `root`, neste kommando vil installer den nybygde glibc til vertssystemet ditt, som mest sannsynlig vil gjøre det ubrukelig. Så dobbeltsjekk at miljøet er riktig innstilt før du kjører følgende kommando.

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

Betydningen av make install alternativene:

`DESTDIR=$LFS`
`DESTDIR` make variabelen brukes av nesten alle pakker for å definere plasseringen der pakken skal være installert. Hvis den ikke er angitt, er den standard til roten (`/`) mappen. Her spesifiserer vi at pakken installeres i `$LFS` , som vil bli roten etter Section 7.4, “Gå inn i Chroot miljøet”.

Fiks hardkodet bane til den kjørbare lasteren i **ldd** skriptet:

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

Caution

På dette tidspunktet er det viktig å stoppe og sikre at de grunnleggende funksjoner (kompilering og lenker) til den nye verktøykjeden fungerer som forventet. For å utføre en tilregnelighetssjekk, kjør følgende kommandoer:

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

Hvis alt fungerer som det skal, skal det ikke være noen feil, og utdata fra den siste kommandoen vil være av formen:

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

Merk at for 32-bits maskiner vil fortolkenavnet være `/lib/ld-linux.so.2`.

Hvis utdataen ikke vises som ovenfor eller det ikke var noen utdata i det hele tatt, da er det noe galt. Undersøk og følg trinnene for å finne ut hvor problemet er og korriger det. Dette problemet må løses før fortsetter.

Når alt er bra, rydd opp i testfilene:

```
rm -v dummy.c a.out
```

Note

Byggepakker i neste kapittel vil fungere som en ekstra sjekk at verktøykjeden er riktig bygget. Hvis noen pakken, spesielt binutils-pass2 eller gcc-pass2, klarer ikke å bygge, er det en indikasjon på at noe har gått galt med tidligere Binutils-, GCC- eller Glibc-installasjoner.

Nå som vår kryssverktøykjede er fullført, fullfør installasjonen limits.h deklarasjoner. For å gjøre det, kjør et verktøy levert av GCC utviklere:

```
$LFS/tools/libexec/gcc/$LFS_TGT/11.2.0/install-tools/mkheaders
```

Detaljer om denne pakken finner du i Section 8.5.3, “Innhold i Glibc.”

5.6. Libstdc++ from GCC-11.2.0, Pass 1

Libstdc++ er standard C++-biblioteket. Det trengs for å kompilere C++-kode (en del av GCC er skrevet i C++), men vi måtte utsette installasjonen da vi bygde gcc-pass1 fordi det avhenger av glibc, som ennå ikke var tilgjengelig i mål mappen.

Omtrentlig byggetid: 0.4 SBU
Nødvendig diskplass: 818 MB

5.6.1. Installation of Target Libstdc++

Note

Libstdc++ er en del av GCC kildene. Du bør først pakke ut GCC tarball og bytte til `gcc-11.2.0` mappen.

Opprett en egen byggemappe for libstdc++ og gå inn i den:

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

Forbered libstdc++ for kompilering:

```
../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++/11.2.0
```

Betydningen av konfigureringsalternativene:

`--host=...`
Spesifiserer at krysskompilatoren vi nettopp har bygget skal brukes i stedet for den i `/usr/bin`.

`--disable-libstdcxx-pch`
Denne bryteren forhindrer installasjon av inkluderte forhåndskompilerte filer som ikke er nødvendige på dette stadiet.

`--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/11.2.0`
Dette spesifiserer installasjonsmappen for inkluderende filer. Fordi libstdc++ er standard C++-biblioteket for LFS, skal denne mappen samsvare med plasseringen der C++-kompilatoren (**$LFS_TGT-g++**) ville søke etter standard C++ inkluderende filer. I en normal konstruksjon, sendes denne informasjonen automatisk til libstdc ++ **configure** alternativer fra toppnivåmappen. I vårt tilfelle, denne informasjonen må gis eksplisitt. C++-kompilatoren vil legge til sysroot banen `$LFS` (spesifisert bygning GCC passerer 1) til inkludere filsøkebanen, så den vil faktisk søke i `$LFS/tools/$LFS_TGT/include/c++/11.2.0`. Kombinasjonen av *`DESTDIR`* variabel (i **make install** kommando nedenfor) og denne bryteren sørger for å installere deklarasjonene der.

Kompiler libstdc++ ved å kjøre:

```
make
```

Installer biblioteket:

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

Detaljer om denne pakken finner du i Section 8.26.2, “Innhold i GCC.”

Chapter 6. Krysskompilering av midlertidige verktøy

6.1. Introduksjon

Dette kapittelet viser hvordan du krysskompilerer grunnleggende verktøy ved å bruke den nettopp bygde kryssverktøykjeden. Disse verktøyene er installert i deres endelige plassering, men kan ikke brukes ennå. Grunnleggende oppgaver er fortsatt avhengige av vertens verktøy. Likevel brukes de installerte bibliotekene ved koblinger.

Bruk av verktøyene vil være mulig i neste kapittel etter å ha gått inn i “chroot” miljøet. Men alle pakkene som bygges i nåværende kapittel må bygges før vi gjør det. Derfor kan vi ikke være uavhengig av vertssystemet ennå.

Nok en gang, la oss huske den feilaktige innstillingen av `LFS` sammen med å bygge som `root`, kan gjøre datamaskinen din ubrukelig. Hele dette kapittelet må gjøres som bruker `lfs`, med miljøet som beskrevet i Section 4.4, “Sette opp miljøet”.

6.2. M4-1.4.19

M4 pakken inneholder en makroprosessor.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 31 MB

6.2.1. Installasjon av M4

Forbered M4 for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.12.2, “Innhold i M4.”

6.3. Ncurses-6.3

Ncurses pakken inneholder biblioteker for terminaluavhengig håndtering av karakterskjermer.

Omtrentlig byggetid: 0.7 SBU
Nødvendig diskplass: 50 MB

6.3.1. Installasjon av Ncurses

Først, sørg for at **gawk** blir funnet først under konfigurasjonen:

```
sed -i s/mawk// configure
```

Kjør deretter følgende kommandoer for å bygge “tic” programmet på byggeverten:

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

Forbered Ncurses for kompilering:

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

Betydningen av de nye konfigureringsalternativene:

`--with-manpage-format=normal`
Dette forhindrer Ncurses fra å installere komprimerte manual sider, noe som kan skje hvis selve vertsdistribusjonen har komprimerte manual sider.

`--without-ada`
Dette sikrer at Ncurses ikke bygger støtte for Ada kompilator som kan være til stede på verten, men som ikke vil være tilgjengelig når vi går inn i **chroot** miljøet.

`--disable-stripping`
Denne bryteren hindrer byggesystemet å strippe programmene som bruker **strip** programmet fra verten. Bruk av vertsverktøy på krysskompilerte program kan forårsake feil.

`--enable-widec`
Denne bryteren forårsaker biblioteker med store tegn (f.eks., `libncursesw.so.6.3`) skal bygges i stedet for vanlige (f.eks., `libncurses.so.6.3`). Disse brede tegnbibliotekene er brukbare i både multibyte og

tradisjonelle 8-biters lokaliteter, mens vanlige biblioteker fungerer som de skal bare i 8-biters lokaliteter. Brede karakterer og normale biblioteker er kildekompatibel, men ikke binærkompatibel.

`--without-normal`
Denne bryteren deaktiverer bygging og installasjon av de fleste statiske biblioteker.

Kompiler pakken:

```
make
```

Installer pakken:

```
make DESTDIR=$LFS TIC_PATH=$(pwd)/build/progs/tic install
echo "INPUT(-lncursesw)" > $LFS/usr/lib/libncurses.so
```

Betydningen av installasjonsalternativene:

`TIC_PATH=$(pwd)/build/progs/tic`
Vi må sende stien til den nettopp bygde **tic** så den er i stand til å kjøre på byggemaskinen, slik at terminaldatabasen kan opprettes uten feil.

echo "INPUT(-lncursesw)" > $LFS/usr/lib/libncurses.so
`libncurses.so` biblioteket trengs av noen få pakker vi skal bygge snart. Vi lager denne lille linkskriptet , da dette er det som gjøres i Chapter 8.

Detaljer om denne pakken finner du i Section 8.28.2, “Innhold i Ncurses.”

6.4. Bash-5.1.16

Bash pakken inneholder Bourne-Again Skallet (Bourne-Again SHell).

Omtrentlig byggetid: 0.4 SBU
Nødvendig diskplass: 64 MB

6.4.1. Installasjon av Bash

Forbered Bash for kompilering:

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

Betydningen av konfigureringsalternativene:

`--without-bash-malloc`
Dette alternativet slår av bruken av Bash minnetildelings funksjon (`malloc`) som er kjent for å forårsake segmenteringsfeil. Ved å slå av dette alternativet vil Bash bruke `malloc` funksjonen fra Glibc som er mer stabil.

Kompiler pakken:

```
make
```

Installer pakken:

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

Lag en lenke for programmene som bruker **sh** til et skall:

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

Detaljer om denne pakken finner du i Section 8.34.2, “Innholdet i Bash.”

6.5. Coreutils-9.0

Pakken Coreutils inneholder verktøy for å vise og stille inn grunnleggende systemegenskaper.

Omtrentlig byggetid: 0.6 SBU
Nødvendig diskplass: 158 MB

6.5.1. Installasjon av Coreutils

Forbered Coreutils for kompilering:

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

Betydningen av konfigureringsalternativene:

`--enable-install-program=hostname`
Dette muliggjør **hostname** binær å bli bygget og installert – den er deaktivert som standard, men kreves av testpakken til Perl.

Kompiler pakken:

```
make
```

Installer pakken:

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

Flytt programmer til deres endelige forventede plasseringer. Selv om dette ikke er nødvendig i dette midlertidige miljøet, må vi gjøre det fordi noen programmer hardkoder kjørbare steder:

```
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
```

Detaljer om denne pakken finner du i Section 8.53.2, “Innhold i Coreutils.”

6.6. Diffutils-3.8

Diffutils pakken inneholder programmer som viser forskjellene mellom filer eller mapper.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 27 MB

6.6.1. Installasjon av Diffutils

Forbered Diffutils for kompilering:

```
./configure --prefix=/usr --host=$LFS_TGT
```

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.55.2, “Innhold i Diffutils.”

6.7. File-5.41

File pakken inneholder et verktøy for å bestemme typen av en gitt fil eller filer.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 32 MB

6.7.1. Installasjon av File

file kommando på byggeverten trenger å være samme versjon som den vi bygger for å opprette signaturfilen. Kjør følgende kommandoer for å bygge den:

```
mkdir build
pushd build
  ../configure --disable-bzlib      \
               --disable-libseccomp \
               --disable-xzlib      \
               --disable-zlib
  make
popd
```

Betydningen av det nye konfigureringsalternativet:

`--disable-*`
Konfigurasjonsskriptet prøver å bruke noen pakker fra vertsdistribusjonen hvis de tilsvarende bibliotekfilene finnes. Det kan føre til kompileringsfeil hvis det finnes en bibliotekfil, men de tilsvarende deklarasjonsfilene ikke gjør det. Disse alternativene forhindrer at det brukes disse unødvendige egenskapene fra verten.

Forbered File for kompilering:

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

Kompiler pakken:

```
make FILE_COMPILE=$(pwd)/build/src/file
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.10.2, “Innholdet i File.”

6.8. Findutils-4.9.0

Findutils pakken inneholder programmer for å finne filer. Disse programmene er gitt for å rekursivt søke gjennom et katalogtre og til å opprette, vedlikeholde og søke i en database (ofte raskere enn den rekursive letingen, men er upålitelig hvis databasen ikke nylig har blitt oppdatert).

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 42 MB

6.8.1. Installasjon av Findutils

Forbered Findutils for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.57.2, “Innhold i Findutils.”

6.9. Gawk-5.1.1

Gawk pakken inneholder programmer for å manipulere tekstfiler.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 45 MB

6.9.1. Installasjon av Gawk

Først, sørg for at noen unødvendige filer ikke blir installert:

```
sed -i 's/extras//' Makefile.in
```

Forbered Gawk for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.56.2, “Innhold i Gawk.”

6.10. Grep-3.7

Grep pakken inneholder programmer for å søke gjennom innholdet i filer.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 26 MB

6.10.1. Installasjon av Grep

Forbered Grep for kompilering:

```
./configure --prefix=/usr   \
            --host=$LFS_TGT
```

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.33.2, “Innhold i Grep.”

6.11. Gzip-1.11

Gzip pakken inneholder programmer for komprimering og dekomprimering av filer.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 11 MB

6.11.1. Installasjon av Gzip

Forbered Gzip for kompilering:

```
./configure --prefix=/usr --host=$LFS_TGT
```

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.60.2, “Contents of Gzip.”

6.12. Make-4.3

Make pakken inneholder et program for å kontrollere genereringen av kjørbare filer og andre ikke-kildefiler av en pakke fra kildefiler.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 15 MB

6.12.1. Installasjon av Make

Forbered Make for kompilering:

```
./configure --prefix=/usr   \
            --without-guile \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Betydningen av det nye konfigureringsalternativet:

`--without-guile`
Selv om vi krysskompilerer, prøver configure å bruke guile fra byggeverten hvis den finner det. Dette gjør at kompileringen mislykkes, så denne bryteren forhindrer bruk av den.

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.64.2, “Innhold i Make.”

6.13. Patch-2.7.6

Patch pakken inneholder et program for å endre eller lage filer ved å bruke en “patch” fil som vanligvis opprettes av **diff** programmet.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 12 MB

6.13.1. Installasjon av Patch

Forbered Patch for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.65.2, “Innhold i oppdateringen.”

6.14. Sed-4.8

Sed pakken inneholder en dataflyt (stream) redigerer.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 20 MB

6.14.1. Installasjon av Sed

Forbered Sed for kompilering:

```
./configure --prefix=/usr   \
            --host=$LFS_TGT
```

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.29.2, “Innhold i Sed.”

6.15. Tar-1.34

Tar pakken gir muligheten til å lage tar arkiver også å utføre forskjellige andre typer arkivmanipulering. Tar kan brukes på tidligere opprettede arkiver for å trekke ut filer, for å lagre flere filer, eller for å oppdatere eller liste filer som allerede var lagret.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 38 MB

6.15.1. Installasjon av Tar

Forbered Tar for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.66.2, “Innhold i Tar.”

6.16. Xz-5.2.5

Xz pakken inneholder programmer for komprimering og dekomprimering av filer. Det gir muligheter for lzma og den nyere xz komprimerings formatene. Komprimering av tekstfiler med **xz** gir en bedre kompresjonsprosent enn med de tradisjonelle **gzip** eller **bzip2** kommandoene.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 15 MB

6.16.1. Installasjon av Xz

Forbered Xz for kompilering:

```
./configure --prefix=/usr                       \
            --host=$LFS_TGT                     \
            --build=$(build-aux/config.guess) \
            --disable-static                    \
            --docdir=/usr/share/doc/xz-5.2.5
```

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.8.2, “Innhold i Xz.”

6.17. Binutils-2.38 - Pass 2

Binutils pakken inneholder en linker, en assembler og annet verktøy for håndtering av objektfiler.

Omtrentlig byggetid: 1.3 SBU
Nødvendig diskplass: 520 MB

6.17.1. Installation of Binutils

Binutils sender en utdatert kopi av libtool i tarballen. Det mangler støtte for sysroot slik at de produserte binærfilene feilaktig kobles til biblioteker fra vertsdistroen. Omgå dette problemet:

```
sed '6009s/$add_dir//' -i ltmain.sh
```

Opprett en egen byggemappe igjen:

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

Forbered Binutils for kompilering:

```
../configure                   \
    --prefix=/usr              \
    --build=$(../config.guess) \
    --host=$LFS_TGT            \
    --disable-nls              \
    --enable-shared            \
    --disable-werror           \
    --enable-64-bit-bfd
```

Betydningen av de nye konfigureringsalternativene:

`--enable-shared`
Bygger `libbfd` som et delt bibliotek.

`--enable-64-bit-bfd`
Aktiverer 64-biters støtte (på verter med smalere ordstørrelser). Kanskje ikke nødvendig på 64-bits systemer, men skader ikke.

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.18.2, “Innhold i Binutils.”

6.18. GCC-11.2.0 - Pass 2

GCC pakken inneholder GNU kompilatorsamlingen, som inkluderer C og C++ kompilatorene.

Omtrentlig byggetid: 11 SBU
Nødvendig diskplass: 3.3 GB

6.18.1. Installasjon av GCC

Som i den første versjonen av GCC, er GMP-, MPFR- og MPC-pakkene nødvendig. Pakk ut tarballene og flytt dem til den nødvendige mappen :

```
tar -xf ../mpfr-4.1.0.tar.xz
mv -v mpfr-4.1.0 mpfr
tar -xf ../gmp-6.2.1.tar.xz
mv -v gmp-6.2.1 gmp
tar -xf ../mpc-1.2.1.tar.gz
mv -v mpc-1.2.1 mpc
```

Hvis du bygger på x86_64, endre standard mappenavn for 64-bit bibliotekene til “lib”:

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

Opprett en egen byggemappe igjen:

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

Lag en symbolkobling som lar libgcc bygges med brukerstøtte for posix tråder:

```
mkdir -pv $LFS_TGT/libgcc
ln -s ../../../libgcc/gthr-posix.h $LFS_TGT/libgcc/gthr-default.h
```

Før du begynner å bygge GCC, husk å deaktivere alle miljø variabler som overstyrer standard optimaliseringsflagg.

Forbered nå GCC for kompilering:

```
../configure                                       \
    --build=$(../config.guess)                     \
    --host=$LFS_TGT                                \
    --prefix=/usr                                  \
    CC_FOR_TARGET=$LFS_TGT-gcc                     \
    --with-build-sysroot=$LFS                      \
    --enable-initfini-array                        \
    --disable-nls                                  \
    --disable-multilib                             \
    --disable-decimal-float                        \
    --disable-libatomic                            \
    --disable-libgomp                              \
    --disable-libquadmath                          \
    --disable-libssp                               \
    --disable-libvtv                               \
    --disable-libstdcxx                            \
    --enable-languages=c,c++
```

Betydningen av de nye konfigureringsalternativene:

`-with-build-sysroot=$LFS`

Normalt bruker *`--host`* å sørge for at en krysskompilator brukes til å bygge GCC, og da vet dennne kompilatoren at den må lete etter overskrifter og biblioteker i `$LFS`. Men byggesystemet til GCC bruker andre verktøy som ikke er klar over denne plasseringen. Denne bryteren sørger for at de finner de nødvendige filene på `$LFS`, og ikke på verten.

`--enable-initfini-array`

Dette alternativet aktiveres automatisk når du bygger en lokal kompilator med en lokal kompilator på x86. Men her bygger vi med en krysskompilator, så vi må eksplisitt angi dette alternativet.

Kompiler pakken:

```
make
```

Installer pakken:

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

Som en siste finpuss kan du lage en symbolkobling. Mange programmer og skript bruker **cc** i stedet for **gcc**, som brukes til å holde programmer generiske og derfor brukbare på alle typer UNIX systemer der GNU C-kompilatoren ikke alltid er installert. Kjøre **cc** lar systemadministratoren bestemme hvilken C-kompilator som skal installeres:

```
ln -sv gcc $LFS/usr/bin/cc
```

Detaljer om denne pakken finner du i Section 8.26.2, “Innhold i GCC.”

Chapter 7. Gå inn i Chroot og bygge ytterligere midlertidige verktøy

7.1. Introduksjon

Dette kapittelet viser hvordan du bygger de siste manglende delene av det midlertidige systemet: verktøyene som trengs for å bygge maskineriet av forskjellige pakker. Nå som alle sirkulære avhengigheter er løst, et "chroot" miljø, fullstendig isolert fra vertsoperativsystemet (bortsett fra den kjørende kjernen), kan brukes til byggingen.

For riktig drift av det isolerte miljøet, noe kommunikasjon med den kjørende kjernen må være etablert. Dette gjøres gjennom det såkalte *Virtuelle kjernefilsystemer (Virtual Kernel File Systems)*, som må være montert når du går inn i chroot miljøet. Det kan være lurt å sjekke at de er montert ved å kjøre **findmnt**.

Før Section 7.4, "Gå inn i Chroot miljøet", kommandoene må kjøres som `root`, med `LFS` variabelsett. Etter å ha gått inn i inn chroot, alle kommandoer kjøres som root, heldigvis uten tilgang til operativsystemet til datamaskinen du bygde LFS på. Vær forsiktig uansett, da det er lett å ødelegge hele LFS system med dårlig utformede kommandoer.

7.2. Skifte eierskap

Note

Kommandoene i resten av denne boken må utføres logget på som bruker `root` og ikke lenger som bruker `lfs`. Også dobbelt sjekk at `$LFS` er satt i `root` sitt miljø.

For øyeblikket er hele mappehierarkiet i `$LFS` eid av brukeren `lfs`, en bruker som bare eksisterer på vertssystemet. Hvis mappene og filene under `$LFS` blir holdt som de er, vil de være eid av en bruker-ID uten en tilsvarende konto. Dette er farlig pga en brukerkonto opprettet senere kan få samme bruker-ID og eie alle filene under `$LFS`, dermed eksponere disse filene til mulig ondsinnet manipulasjon.

For å løse dette problemet, endre eierskap til `$LFS/*` mapper til bruker `root` ved å kjøre følgende kommando:

```
chown -R root:root $LFS/{usr,lib,var,etc,bin,sbin,tools}
case $(uname -m) in
  x86_64) chown -R root:root $LFS/lib64 ;;
esac
```

7.3. Klargjøring av virtuelle kjernefilsystemer

Ulike filsystemer eksportert av kjernen brukes til å kommunisere til og fra selve kjernen. Disse filsystemene er virtuelle ved at ingen disk plass brukes til dem. Innholdet i filsystemene ligger i minnet.

Begynn med å lage mapper som filsystemene vil bli montert på:

```
mkdir -pv $LFS/{dev,proc,sys,run}
```

7.3.1. Opprette innledende enhetsnoder

Når kjernen starter opp systemet, krever det tilstedeværelse av noen få enhetsnoder, spesielt `console` og `null` enheter. Enhetsnodene må være opprettet på harddisken slik at de er tilgjengelige før kjernen fyller `/dev`), og i tillegg når Linux startes med *`init=/bin/bash`*. Opprett enhetene ved å kjøre følgende kommandoer:

```
mknod -m 600 $LFS/dev/console c 5 1
mknod -m 666 $LFS/dev/null c 1 3
```

7.3.2. Montering og fylling av /dev

Den anbefalte metoden for å fylle `/dev` mappen med enheter er å montere et virtuelt filsystem (som f.eks `tmpfs`) på `/dev` mappen, og la enhetene være opprettet dynamisk på det virtuelle filsystemet etter hvert som de oppdages eller gitt tilgang til. Enhetsopprettingen gjøres vanligvis under oppstartsprosessen av Udev. Siden dette nye systemet ennå ikke har Udev og har ennå ikke blitt startet opp, er det nødvendig å montere og fylle `/dev` manuelt. Dette oppnås ved å binde montering av vertssystemets `/dev` mappen. Et bind mount er en spesiell type montering som lar deg lage et speil av en mappe eller monteringspunkt til et annet sted. Bruk følgende kommando for å oppnå dette:

```
mount -v --bind /dev $LFS/dev
```

7.3.3. Montering av virtuelle kjernefilsystemer

Monter nå de gjenværende virtuelle kjernefilsystemene:

```
mount -v --bind /dev/pts $LFS/dev/pts
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
mount -vt tmpfs tmpfs $LFS/run
```

I noen vertssystemer, `/dev/shm` er en symbolsk lenke til `/run/shm`. /run tmpfs ble montert ovenfor, så i dette tilfellet er det bare en mappe som må opprettes.

```
if [ -h $LFS/dev/shm ]; then
  mkdir -pv $LFS/$(readlink $LFS/dev/shm)
fi
```

7.4. Gå inn i Chroot miljøet

Nå som alle pakkene som kreves for å bygge resten av nødvendige verktøy er på systemet, er det på tide å gå inn i chroot miljøet for å fullføre installasjonen av de gjenværende midlertidige verktøyene. Dette miljøet vil også brukes for å installere det endelige systemet. Som bruker `root`, kjør følgende kommando for å gå inn i miljøet som for øyeblikket er befolket med bare midlertidige verktøy:

```
chroot "$LFS" /usr/bin/env -i   \
    HOME=/root                  \
    TERM="$TERM"                \
    PS1='(lfs chroot) \u:\w\$ ' \
    PATH=/usr/bin:/usr/sbin     \
    /bin/bash --login
```

-i alternativet gitt til **env** kommandoen vil slette alle variabler i chroot miljøet. Etter det, bare HOME, TERM, PS1, og PATH variablene settes på nytt. *TERM=$TERM* konstruksjonen vil sette TERM variabelen inne i chroot til samme verdi som utenfor chroot. Denne variabelen er nødvendig for programmer som **vim** og **less** å fungere skikkelig. Hvis andre variabler ønskes, som f.eks CFLAGS eller CXXFLAGS, dette er et bra sted å sette dem igjen.

Fra dette tidspunktet er det ikke nødvendig å bruke LFS variabelen lenger fordi alt arbeid vil være begrenset til LFS filsystemet. Dette er fordi Bash skallet blir fortalt at for $LFS er roten nå (/) mappen.

Legg merke til at /tools/bin ikke er i PATH. Dette betyr at kryssverktøykjeden ikke lenger vil være det som brukes i chroot miljøet.

Merk at ledeteksten til **bash** vil si I have no name! Dette er normalt fordi /etc/passwd filen ikke er opprettet ennå.

Note

Det er viktig at alle kommandoene gjennom resten av dette kapittel og de følgende kapitlene kjøres fra chroot miljøet. Hvis du forlater dette miljøet av en eller annen grunn (omstart for eksempel), sørg for at de virtuelle kjernefilsystemene er montert som forklart i Section 7.3.2, “Montering og fylling av /dev” og Section 7.3.3, “Montering av virtuelle kjernefilsystemer” og gå inn i chroot igjen før du fortsetter med installasjonen.

7.5. Opprette mapper

Det er på tide å lage hele strukturen i LFS filsystemet.

Lag noen mapper på rotnivå som ikke er i det begrensede settet som kreves i de foregående kapitlene ved å gi følgende kommando:

Note

Noen av mappene nedenfor er allerede opprettet med eksplisitte instruksjoner eller når du installerte noen pakker. De gjentas nedenfor for fullstendighet.

```
mkdir -pv /{boot,home,mnt,opt,srv}
```

Lag det nødvendige settet med undermapper under rotnivået ved å utstede følgende kommandoer:

```
mkdir -pv /etc/{opt,sysconfig}
mkdir -pv /lib/firmware
mkdir -pv /media/{floppy,cdrom}
mkdir -pv /usr/{,local/}{include,src}
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
```

mapper er som standard opprettet med tillatelsesmodus 755, men dette er ikke ønskelig for alle mapper. I kommandoene ovenfor, to endringer gjøres—en til `root` brukerens hjemmemappe, og en annen til mappene for midlertidige filer.

Den første modusendringen sikrer at ikke hvem som helst kan komme inn `/root` mappen—det samme som en vanlig bruker ville gjort med sin hjemmemappe. De andre modusendring sørger for at enhver bruker kan skrive til `/tmp` og `/var/tmp` mapper, men kan ikke fjernes en annen brukers filer fra dem. Sistnevnte er forbudt av den såkalte “låst bit (sticky bit),” den høyeste biten (1) i 1777 bitmasken.

7.5.1. FHS Samsvarsmerknad

mappetreet er basert på Filsystemhierarkistandard (Filesystem Hierarchy Standard) (FHS) (tilgjengelig på *https://refspecs.linuxfoundation.org/fhs.shtml*). FHS spesifiserer også den valgfrie tilstedeværelsen av noen mapper som f.eks `/usr/local/games` og `/usr/share/games`. Vi skaper kun mapper som trengs. Du må imidlertid gjerne lage disse mappene.

7.6. Opprette essensielle filer og symbolkoblinger

Historisk sett har Linux en liste over de monterte filsystemene i filen `/etc/mtab`. Moderne kjerner opprettholder denne listen internt og eksponerer det for brukeren via `/proc` filsystemet. For å tilfredsstille verktøy som forventer tilstedeværelse av `/etc/mtab`, opprett følgende symbolske lenke:

```
ln -sv /proc/self/mounts /etc/mtab
```

Lag en grunnleggende `/etc/hosts` fil som blr referert til i noen testsuiter, og også i en av Perls konfigurasjonsfiler :

```
cat > /etc/hosts << EOF
127.0.0.1  localhost $(hostname)
::1        localhost
EOF
```

For at bruker `root` skal kunne logge inn og for navnet “root” å bli gjenkjent, det må være relevante oppføringer i `/etc/passwd` og `/etc/group` filene.

Opprett `/etc/passwd` filen ved å kjøre følgende kommando:

```
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
systemd-journal-gateway:x:73:73:systemd Journal Gateway:/:/usr/bin/false
systemd-journal-remote:x:74:74:systemd Journal Remote:/:/usr/bin/false
systemd-journal-upload:x:75:75:systemd Journal Upload:/:/usr/bin/false
systemd-network:x:76:76:systemd Network Management:/:/usr/bin/false
systemd-resolve:x:77:77:systemd Resolver:/:/usr/bin/false
systemd-timesync:x:78:78:systemd Time Synchronization:/:/usr/bin/false
systemd-coredump:x:79:79:systemd Core Dumper:/:/usr/bin/false
uuidd:x:80:80:UUID Generation Daemon User:/dev/null:/usr/bin/false
systemd-oom:x:81:81:systemd Out Of Memory Daemon:/:/usr/bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/usr/bin/false
EOF
```

Selve passordet for `root` velges senere.

Opprett `/etc/group` filen ved å kjøre følgende kommando:

```
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:
usb:x:14:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
systemd-journal:x:23:
input:x:24:
mail:x:34:
kvm:x:61:
systemd-journal-gateway:x:73:
systemd-journal-remote:x:74:
systemd-journal-upload:x:75:
systemd-network:x:76:
systemd-resolve:x:77:
systemd-timesync:x:78:
systemd-coredump:x:79:
uuidd:x:80:
systemd-oom:x:81:
wheel:x:97:
nogroup:x:99:
users:x:999:
EOF
```

De opprettede gruppene er ikke en del av noen standard—de er grupper delvis bestemt av kravene til Udev konfigurasjonen i kapittel 9, og delvis etter felles konvensjon brukt av en rekke eksisterende Linux distribusjoner. I tillegg er noen testsuiter avhengige av spesifikke brukere eller grupper. Linux Standard Base (LSB, tilgjengelig på *http://refspecs.linuxfoundation.org/lsb.shtml*) anbefaler bare at, foruten gruppen `root` med en Gruppe ID (GID) på 0, en gruppe `bin` med en GID på 1 å være tilstede. Alle andre gruppenavn og GID-er kan velges fritt av systemadministratoren siden velskrevne programmer ikke er avhengige av GID-nummer, men bruk heller gruppens navn.

Noen tester i Chapter 8 trenger en vanlig bruker. Vi legger til denne brukeren her og sletter denne kontoen på slutten av det kapittelet.

```
echo "tester:x:101:101::/home/tester:/bin/bash" >> /etc/passwd
echo "tester:x:101:" >> /etc/group
install -o tester -d /home/tester
```

For å fjerne “I have no name!” ledetekst, start et nytt skall. Siden `/etc/passwd` og `/etc/group` filer har blitt opprettet, vil brukernavn og gruppenavnoppløsning nå virke:

```
exec /usr/bin/bash --login
```

login, **agetty**, og **init** programmene (og andre) bruker en rekke logg filer for å registrere informasjon som hvem som var logget inn på systemet og når. Disse programmene vil imidlertid ikke skrive til loggfilene hvis de ikke allerede eksisterer. Initialiser loggfilene og gi dem riktige tillatelser:

```
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
```

`/var/log/wtmp` filen registrerer alle påloggінger og logger ut. `/var/log/lastlog` filen registrerer når hver bruker sist logget på. `/var/log/faillog` filen registrerer mislykkede påloggingsforsøk. `/var/log/btmp` filen registrerer de dårlige påloggingsforsøkene.

Note

`/run/utmp` filen registrerer brukerne som for øyeblikket er pålogget. Denne filen opprettes dynamisk i oppstart skriptet.

7.7. Libstdc++ from GCC-11.2.0, Pass 2

Ved bygging av gcc-pass2 måtte vi utsette installasjonen av C++ standardbiblioteket fordi ingen passende kompilator var tilgjengelig for å kompilere den. Vi kunne ikke bruke den innebygde kompilatoren fordi det er en innebygd kompilator og ikke bør brukes utenfor chroot og risikere å forurense bibliotekene med enkelte vertskomponenter.

Omtrentlig byggetid: 0.8 SBU
Nødvendig diskplass: 1.1 GB

7.7.1. Installasjon av Target Libstdc++

Note

Libstdc++ er en del av GCC kildene. Du bør først pakke ut GCC tarballen og bytte til `gcc-11.2.0` mappen.

Lag en lenke som eksisterer når du bygger libstdc++ i gcc treet:

```
ln -s gthr-posix.h libgcc/gthr-default.h
```

Opprett en egen byggemappe for libstdc++ og skriv gå inn i den:

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

Forbered libstdc++ for kompilering:

```
../libstdc++-v3/configure           \
    CXXFLAGS="-g -O2 -D_GNU_SOURCE"  \
    --prefix=/usr                   \
    --disable-multilib              \
    --disable-nls                   \
    --host=$(uname -m)-lfs-linux-gnu \
    --disable-libstdcxx-pch
```

Betydningen av konfigureringsalternativene:

`CXXFLAGS="-g -O2 -D_GNU_SOURCE"`
Disse flaggene sendes av Makefile på øverste nivå når det bygges en komplett versjon av GCC.

`--host=$(uname -m)-lfs-linux-gnu`
Vi må etterligne hva som ville skje hvis denne pakken ble bygget som en del av et komplett kompilatorbygg. Denne bryteren vil bli sendt til configure av GCCs byggemaskineri.

`--disable-libstdcxx-pch`
Denne bryteren forhindrer installasjon av forhåndskompilerte inkluderte filer som ikke er nødvendige på dette stadiet.

Kompiler libstdc++ ved å kjøre:

```
make
```

Installer biblioteket:

```
make install
```

Detaljer om denne pakken finner du i Section 8.26.2, “Innhold i GCC.”

7.8. Gettext-0.21

Gettext pakken inneholder verktøy for internasjonalisering og lokalisering. Disse gjør at programmer kan kompileres med NLS (Lokal Språk Støtte), slik at de kan sende ut meldinger i brukerens lokale språkformat.

Omtrentlig byggetid: 1.6 SBU
Nødvendig diskplass: 280 MB

7.8.1. Installasjon av Gettext

For vårt midlertidige sett med verktøy trenger vi bare å installere tre programmer fra Gettext.

Forbered Gettext for kompilering:

```
./configure --disable-shared
```

Betydningen av konfigureringsalternativet:

`--disable-shared`
Vi trenger ikke å installere noen av de delte Gettext bibliotekene, denne gangen er det derfor ikke nødvendig å bygge dem.

Kompiler pakken:

```
make
```

Installer **msgfmt**, **msgmerge**, og **xgettext** programmene:

```
cp -v gettext-tools/src/{msgfmt,msgmerge,xgettext} /usr/bin
```

Detaljer om denne pakken finner du i Section 8.31.2, “Innhold i Gettext.”

7.9. Bison-3.8.2

Bison pakken inneholder en parsergenerator.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 50 MB

7.9.1. Installasjon av Bison

Forbered Bison for kompilering:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/bison-3.8.2
```

Betydningen av det nye konfigureringsalternativet:

`--docdir=/usr/share/doc/bison-3.8.2`
Dette forteller byggesystemet å installere bison dokumentasjonen i en versjonert mappe.

Kompiler pakken:

```
make
```

Installer pakken:

```
make install
```

Detaljer om denne pakken finner du i Section 8.32.2, “Innholdet i Bison.”

7.10. Perl-5.34.0

Perl pakken inneholder den praktiske utvinnings og rapporterings språket (Practical Extraction and Report Language).

Omtrentlig byggetid: 1.6 SBU
Nødvendig diskplass: 272 MB

7.10.1. Installasjon av Perl

Forbered Perl for kompilering:

```
sh Configure -des                                         \
             -Dprefix=/usr                                \
             -Dvendorprefix=/usr                          \
             -Dprivlib=/usr/lib/perl5/5.34/core_perl      \
             -Darchlib=/usr/lib/perl5/5.34/core_perl      \
             -Dsitelib=/usr/lib/perl5/5.34/site_perl      \
             -Dsitearch=/usr/lib/perl5/5.34/site_perl     \
             -Dvendorlib=/usr/lib/perl5/5.34/vendor_perl \
             -Dvendorarch=/usr/lib/perl5/5.34/vendor_perl
```

Betydningen av de nye konfigureringsalternativene:

`-des`
Dette er en kombinasjon av tre alternativer: -d bruker standardinnstillinger for alle elementer; -e sikrer gjennomføring av alle oppgaver; -s sender ikke ut ikke-essensielle utdata.

Kompiler pakken:

```
make
```

Installer pakken:

```
make install
```

Detaljer om denne pakken finner du i Section 8.41.2, “Innhold i Perl.”

7.11. Python-3.10.2

Python 3 pakken inneholder Python utviklingsmiljøet. Den er nyttig for objektorientert programmering, skriving av skript, prototyping store programmer, eller utvikle hele applikasjoner.

Omtrentlig byggetid: 1.2 SBU
Nødvendig diskplass: 359 MB

7.11.1. Installasjon av Python

Note

Det er to pakkefiler som navnet begynner med “python”. Den å pakke ut er `Python-3.10.2.tar.xz` (legg merke til stor bokstav først).

Forbered Python for kompilering:

```
./configure --prefix=/usr   \
            --enable-shared \
            --without-ensurepip
```

Betydningen av konfigureringsalternativet:

`--enable-shared`
Denne bryteren forhindrer installasjon av statiske biblioteker.

`--without-ensurepip`
Denne bryteren deaktiverer Python pakkeinstallasjonsprogrammet, som ikke er nødvendig på dette stadiet.

Kompiler pakken:

```
make
```

Note

Noen Python 3 moduler kan ikke bygges nå på grunn av avhengighetene ikke er installert ennå. Byggesystemet prøver imidlertid å bygge dem, så kompileringen av noen filer vil mislykkes og kompilatormeldingen kan synes å indikere “fatal error”. Meldingen bør ignoreres. Bare sørg for toppnivåets **make** kommando ikke har feilet. De valgfrie modulene er ikke nødvendig nå, og de vil bli bygget i Chapter 8.

Installer pakken:

```
make install
```

Detaljer om denne pakken finner du i Section 8.50.2, “Innhold i Python 3.”

7.12. Texinfo-6.8

Texinfo pakken inneholder programmer for lesing, skriving og konvertere informasjonssider.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 109 MB

7.12.1. Installasjon av Texinfo

Først, fiks et problem med å bygge pakken med Glibc-2.34 eller nyere:

```
sed -e 's/__attribute_nonnull__/__nonnull/' \
    -i gnulib/lib/malloc/dynarray-skeleton.c
```

Forbered Texinfo for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

```
make install
```

Detaljer om denne pakken finner du i Section 8.67.2, “Innhold i Texinfo.”

7.13. Util-linux-2.37.4

Util-linux pakken inneholder diverse hjelpeprogrammer.

Omtrentlig byggetid: 0.7 SBU
Nødvendig diskplass: 129 MB

7.13.1. Installasjon av Util-linux

FHS anbefaler å bruke `/var/lib/hwclock` mappen i stedet for den vanlige `/etc` mappen som plassering for `adjtime` filen. Opprett denne mappen med:

```
mkdir -pv /var/lib/hwclock
```

Forbered Util-linux for kompilering:

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime    \
            --libdir=/usr/lib    \
            --docdir=/usr/share/doc/util-linux-2.37.4 \
            --disable-chfn-chsh  \
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
            --disable-runuser    \
            --disable-pylibmount \
            --disable-static     \
            --without-python     \
            runstatedir=/run
```

Betydningen av konfigureringsalternativene:

`ADJTIME_PATH=/var/lib/hwclock/adjtime`
Dette angir plasseringen av filopptaksinformasjonen om maskinvareklokken i henhold til FHS. Dette er ikke strengt tatt nødvendig for dette midlertidige verktøyet, men det forhindrer at det lages en fil på et annet sted, som ikke ville bli overskrevet eller fjernet når du bygger den endelige util-linux pakken.

`--libdir=/usr/lib`
Denne bryteren sikrer `.so` målrettes mot symbolkoblinger i den delte bibliotekfilen i samme mappe (`/usr/lib`) direkte.

`--disable-*`
Disse bryterne forhindrer advarsler om bygningskomponenter som krever pakker som ikke er i LFS eller ikke er installert ennå.

`--without-python`
Denne bryteren deaktiverer bruk av Python. Den unngår å prøve å bygge unødvendige bindinger.

`runstatedir=/run`
Denne bryteren angir plasseringen av socket som brukes av **uuidd** og `libuuid` riktig.

Kompiler pakken:

```
make
```

Installer pakken:

```
make install
```

Detaljer om denne pakken finner du i Section 8.75.2, “Innhold i Util-linux.”

7.14. Opprydding og lagring av det midlertidige systemet

7.14.1. Opprydding

Fjern først den installerte dokumentasjonen for å forhindre dem fra å havne i det endelige systemet, og å spare ca 35 MB:

```
rm -rf /usr/share/{info,man,doc}/*
```

For det andre er .la filene til libtool bare nyttige når du kobler til statiske biblioteker. De er unødvendige og potensielt skadelige ved bruk av dynamiske delte biblioteker, spesielt når du bruker byggesystemer som ikke er autoverktøy. Mens du fortsatt er i chroot, fjern disse filene nå:

```
find /usr/{lib,libexec} -name \*.la -delete
```

Den nåværende systemstørrelsen er nå omtrent 3 GB, /tools mappen er ikke lenger nødvendig. Den bruker ca 1 GB diskplass. Slett den nå:

```
rm -rf /tools
```

7.14.2. Sikkerhetskopiering

På dette tidspunktet er de essensielle programmene og bibliotekene opprettet og ditt nåværende LFS system er i god stand. Systemet ditt kan nå bli sikkerhetskopiert for senere gjenbruk. Ved fatale feil i de påfølgende kapitler, viser det seg ofte at å fjerne alt og starte på nytt (mer forsiktig) er det beste alternativet for å gjenopprette. Dessverre, alle midlertidige filer vil også bli fjernet. For å unngå å bruke ekstra tid på gjøre om noe som har blitt bygget vellykket, og lage en sikkerhetskopi av det nåværende LFS systemet kan vise seg å være nyttig.

Note

Alle de resterende trinnene i denne delen er valgfrie. Likevel, så snart du begynner å installere pakker i Chapter 8, vil de midlertidige filene bli overskrevet. Så det kan være lurt å ta en sikkerhetskopi av systemet som beskrevet nedenfor.

Følgende trinn utføres fra utenfor chroot miljøet. Det betyr at du må forlate chroot miljøet før du fortsetter. Grunnen til det er å få tilgang til filsystemplasseringer utenfor chroot miljøet for å lagre/lese sikkerhetskopiarkivet som ikke skal plasseres innenfor `$LFS` hierarkiet for sikkerhetsmessige årsaker.

Hvis du har bestemt deg for å ta en sikkerhetskopi, forlat chroot miljøet:

```
exit
```

Important

Alle de følgende instruksjonene utføres av `root` på vertssystemet ditt. Vær ekstra forsiktig med kommandoene du skal kjøre ettersom feil her kan du endre vertssystemet ditt. Vær oppmerksom på at miljøvariabelen `LFS` er satt for bruker `lfs` som standard er kanskje *ikke* satt for `root`.

Når kommandoer skal utføres av `root`, sørg for at du har satt `LFS`.

Dette har vært diskutert i Section 2.6, “Stille inn $LFS variabelen”.

Før du lager en sikkerhetskopi, avmonter de virtuelle filsystemene:

```
umount $LFS/dev/pts
umount $LFS/{sys,proc,run,dev}
```

Sørg for at du har minst 1 GB ledig diskplass (kildenes tarballer vil bli inkludert i sikkerhetskopiarkivet) på filsystemet som inneholder mappen der du oppretter sikkerhetskopiarkivet.

Merk at instruksjonene nedenfor spesifiserer hjemmemappen til vertenssystemets bruker `root` som vanligvis finnes på rotfilsystemet.

Erstatt `$HOME` av en mappe etter eget valg hvis du ikke ønsker å ha sikkerhetskopien lagret i `root` sin hjemmemappe.

Opprett sikkerhetskopiarkivet ved å kjøre følgende kommando:

Note

Fordi sikkerhetskopieringsarkivet er komprimert, tar det relativt lang tid lang tid (over 10 minutter) selv på et rimelig raskt system.

```
cd $LFS
tar -cJpf $HOME/lfs-temp-tools-11.1-systemd.tar.xz .
```

Note

Hvis du fortsetter til kapittel 8, ikke glem å gå inn i chroot miljøet på nytt som forklart i “Viktig” boksen under.

7.14.3. Gjenopprett

I tilfelle noen feil har blitt gjort og du må begynne på nytt, kan du bruk denne sikkerhetskopien til å gjenopprette systemet og spare litt gjenopprettingstid. Siden kildene ligger under `$LFS`, er de inkludert i sikkerhetskopieringsarkivet, slik at de ikke trenger å lastes ned igjen. Etter å ha sjekket at `$LFS` er riktig innstilt, gjenopprett sikkerhetskopien ved å utføre følgende kommandoer:

Warning

Følgende kommandoer er ekstremt farlige. Hvis du kjører **rm -rf ./*** som root brukeren og du ikke endret til $LFS-mappen eller `LFS` miljøvariabelen ikke er satt for brukeren root, vil den ødelegge hele vertssystemet ditt. DU ER ADVART.

```
cd $LFS
rm -rf ./*
tar -xpf $HOME/lfs-temp-tools-11.1-systemd.tar.xz
```

Igjen, dobbeltsjekk at miljøet er riktig konfigurert og fortsett å bygge resten av systemet.

Important

Hvis du forlot chroot-miljøet for å lage en sikkerhetskopi eller starte byggingen på nytt ved hjelp av en gjenoppretting, husk å sjekke at det virtuelle filsystemer fortsatt er montert (**findmnt | grep $LFS**). Hvis de ikke er montert, monter dem på nytt nå som beskrevet i Section 7.3, “Klargjøring av virtuelle kjernefilsystemer” og gå inn i chroot miljøet igjen (se Section 7.4, “Gå inn i Chroot miljøet”) før du fortsetter.

Part IV. Bygge LFS systemet

Chapter 8. Installere grunnleggende systemprogramvare

8.1. Introduksjon

I dette kapittelet begynner vi for alvor å bygge LFS systemet.

Installasjonen av denne programvaren er enkel. Skjønt i mange tilfeller kan installasjonsinstruksjonene gjøres kortere og mer generelle, vi har valgt å gi de fullstendige instruksjonene for hver pakke for å minimere mulighetene for feil. Nøkkelen til å lære hva som gjør et at et Linux system virker er å vite hva hver pakke brukes til og hvorfor du (eller systemet) kan trenge det.

Vi anbefaler ikke å bruke optimaliseringer. De kan gjøre at et program kjører litt raskere, men de kan også forårsake kompilerings vanskeligheter og problemer når du kjører programmet. Hvis en pakke nekter å kompilere når du bruker optimalisering, prøv å kompilere den uten optimalisering og se om det løser problemet. Selv om pakken kompileres ved bruk av optimalisering, er det risiko for at det kan ha blitt kompilert feil fordi de komplekse interaksjonene mellom koden og byggeverktøyene. Legg også merke til at `-march` og `-mtune` alternativene som ikke er spesifisert i boken er ikke testet. Dette kan skape problemer med verktøykjedepakkene (Binutils, GCC og Glibc). De små potensielle gevinstene oppnådd ved bruk av kompilatoroptimaliseringer oppveies ofte av risikoen. Førstegangsbyggere av LFS oppfordres til å bygge uten tilpassete optimaliseringer. Det påfølgende systemet vil fortsatt kjøre veldig raskt og være stabilt samtidig.

Før installasjonsinstruksjonene gir hver installasjonsside informasjon om pakken, inkludert en kortfattet beskrivelse av hva den inneholder, omtrent hvor lang tid det vil ta å bygge, og hvor mye diskplass kreves under denne byggeprosessen. Etter installasjoneninstruksene , er det en liste over programmer og biblioteker (sammen med korte beskrivelser) som pakken installerer.

Note

SBU verdiene og nødvendig diskplass inkluderer testpakkedata for alle gjeldende pakker i Chapter 8. SBU verdier har blitt beregnet ved å bruke en enkelt CPU-kjerne (-j1) for alle operasjoner.

8.1.1. Om biblioteker

Generelt fraråder LFS redaktørene å bygge og installere statiske biblioteker. Det opprinnelige formålet for de fleste statiske biblioteker er nå foreldet i et moderne Linux system. I tillegg kan koblinger for et statiske biblioteker i et program være skadelig. Hvis en oppdatering til biblioteket er nødvendig for å fjerne et sikkerhetsproblem, må alle programmer som bruker det statiske biblioteket kobles til det nye biblioteket. Siden bruken av statiske biblioteker ikke alltid er åpenbart, de relevante programmene (og prosedyrene som trengs for å gjør koblingen) er kanskje ikke engang kjent.

I prosedyrene i dette kapittelet fjerner eller deaktiverer vi installasjon av de fleste statiske biblioteker. Vanligvis gjøres dette ved å utstede en `--disable-static` alternativ til **configure**. I andre tilfeller er det nødvendig med alternative midler. I noen få tilfeller, spesielt glibc og gcc, forblir bruken av statiske biblioteker avgjørende for det generelle pakke byggeprosess.

For en mer fullstendig diskusjon av biblioteker, se diskusjonen *Biblioteker: Statiske eller delte?* i BLFS boken.

8.2. Pakkehåndtering

Pakkebehandling er et ofte etterspurt tillegg til LFS boken. En pakkebehandler lar deg spore installasjonen av filer som gjør det enkelt å fjerne og oppgradere pakker. I tillegg til binær- og biblioteksfilene, en pakkebehandler vil håndtere installasjonen av konfigurasjonsfiler. Før du begynner å lure på, NEI—denne delen vil ikke snakke om eller anbefale noen spesiell pakkebehandler. Det den gir er en oppsummering av de fleste populære teknikker og hvordan de fungerer. Den perfekte pakkebehandleren for deg vil kanskje være blant disse teknikkene eller kan være en kombinasjon av to eller flere av disse teknikker. Denne delen nevner kort problemer som kan oppstå ved oppgradering av pakker.

Noen grunner til at ingen pakkebehandler er nevnt i LFS eller BLFS inkludere:

- Å håndtere pakkehåndtering fjerner fokus fra målene til disse bøkene—som lærer hvordan et Linux system er bygget.
- Det er flere løsninger for pakkehåndtering, som hver har dens styrker og ulemper. Inkludere en som tilfredsstiller alle målgrupper er vanskelig.

Det er skrevet noen tips om emnet pakkehåndtering. Besøk *Hints Project* og se om en av dem passer ditt behov.

8.2.1. Oppgraderingsproblemer

En Pakkehåndterer gjør det enkelt å oppgradere til nyere versjoner når de er utgitt. Generelt kan instruksjonene i LFS- og BLFS-bøkene brukes til å oppgradere til nyere versjoner. Her er noen punkter du bør være oppmerksom på når du oppgraderer pakker, spesielt på et kjørende system.

- Hvis Linux kjernen må oppgraderes (for eksempel fra 5.10.17 til 5.10.18 eller 5.11.1), må ikke noe annet bygges om. Systemet vil fortsette å fungere bra takket være den veldefinerte grensen mellom kjernen og brukerområdet. Nærmere bestemt Linux API-deklarasjoner trenger ikke å (og bør ikke bli, se neste element) oppgraderes ved siden av kjernen. Du må starte systemet på nytt for å bruke den oppgraderte kjernen.
- Hvis Linux API-deklarasjoner eller Glibc må oppgraderes til en nyere versjon, (f.eks. fra glibc-2.31 til glibc-2.32), er det tryggere å gjenoppbygge LFS. Selv om du *kanskje* kan gjenoppbygge alle pakkene i deres avhengighetsrekkefølge, anbefaler vi ikke den.
- Hvis en pakke som inneholder et delt bibliotek oppdateres, og hvis navnet på biblioteket endres, vil eventuelle pakker dynamisk koblet til biblioteket måtte kompileres på nytt for å kunne kobles mot det nyere biblioteket. (Merk at det ikke er noen sammenheng mellom pakkeversjon og navnet på biblioteket.) Tenk for eksempel på en pakke foo-1.2.3 som installerer et delt bibliotek med navn `libfoo.so.1`. Hvis du oppgraderer pakken til en nyere versjon foo-1.2.4 som installerer et delt bibliotek med navn `libfoo.so.2`. I dette tilfellet, alle pakker som er dynamisk koblet til `libfoo.so.1` må kompileres på nytt for å lenke imot `libfoo.so.2` for å bruke den nye bibliotekversjonen. Du bør ikke fjerne den forrige biblioteker med mindre alle de avhengige pakkene er rekompilert.
- Hvis en pakke som inneholder et delt bibliotek oppdateres, og navnet på biblioteket ikke endres, men versjonsnummeret til biblioteket **file** reduseres (f.eks. navnet på biblioteket beholdes ved navn `libfoo.so.1`, men navnet på bibliotekfilen er endret fra `libfoo.so.1.25` til `libfoo.so.1.24`), bør du fjerne bibliotekfilen fra den tidligere installerte versjonen (`libfoo.so.1.25` i dette tilfellet). Eller, et **ldconfig** kjør (selv ved å bruke en kommando linje, eller ved installasjon av en pakke) vil tilbakestille symbolkoblingen `libfoo.so.1` til å peke på den gamle bibliotekfilen fordi den ser ut til å ha en “nyere” versjon, ettersom versjonsnummeret er større. Denne situasjonen kan skje hvis du må nedgradere en pakke, ellers endrer pakken versjonen ordningen med bibliotekfiler plutselig.

- Hvis en pakke som inneholder et delt bibliotek oppdateres, og navnet på biblioteket ikke endres, men et alvorlig problem (spesielt en sikkerhetssårbarhet) er fikset, alle programmer som kjører koblet til det delte biblioteket bør startes på nytt. Følgende kommando, kjørt som `root` etter oppdateringen, vil liste opp hva som bruker de gamle versjonene av disse bibliotekene (erstatt *`libfoo`* med navnet på biblioteket):

```
grep -l  -e 'libfoo.*deleted' /proc/*/maps |
   tr -cd 0-9\\n | xargs -r ps u
```

  Hvis OpenSSH brukes for tilgang til systemet og det er koblet til det oppdaterte biblioteket, må du omstarte **sshd** tjeneste, deretter logg ut, logg på igjen, og kjør kommandoen på nytt for å bekrefte at ingenting fortsatt bruker de slettede bibliotekene.

- Hvis et binært eller et delt bibliotek overskrives, kan prosessene som bruker koden eller dataene i binærfilen eller biblioteket krasje. Den riktige måten å oppdatere et binært eller et delt bibliotek uten å forårsake at prosessen krasjer er å fjerne den først, og deretter installere den nye versjonen. **install** kommandoen levert av Coreutils har allerede implementert dette og de fleste pakker bruker det til å installere binærfiler og biblioteker. Dette betyr at du ikke vil bli plaget av dette problemet mesteparten av tiden. Imidlertid er installasjonsprosessen for noen pakker (spesielt Mozilla JS i BLFS) bare å overskrive filen hvis den eksisterer og forårsaker et krasj, så det er tryggere å lagre arbeidet ditt og lukke unødvendige kjørende prosesser før du oppdaterer en pakke.

8.2.2. Pakkehåndteringsteknikker

Følgende er noen vanlige pakkehåndteringsteknikker. Før du ta en avgjørelse om en pakkeforvalter, gjør litt research på de forskjellige teknikker, spesielt ulempene ved den spesielle ordningen.

8.2.2.1. Det er alt i hodet mitt!

Ja, dette er en pakkehåndteringsteknikk. Noen mennesker finner ikke behovet for en pakkehåndterer fordi de kjenner pakkene inngående og vet hvilke filer som er installert av hver pakke. Noen brukere trenger heller ikke pakkehåndtering fordi de planlegger å gjenoppbygge hele system når en pakke endres.

8.2.2.2. Installer i separate kataloger

Dette er en forenklet pakkehåndtering som ikke trenger noe ekstra pakke for å administrere installasjonene. Hver pakke er installert i en egen katalog. For eksempel er pakke foo-1.1 installert i `/usr/pkg/foo-1.1` og en symbolkobling er laget fra `/usr/pkg/foo` til `/usr/pkg/foo-1.1`. Ved installasjon en ny versjon foo-1.2, installeres den i `/usr/pkg/foo-1.2` og den forrige symbolkoblingen erstattes av en symbolkobling til den nye versjonen.

Miljøvariabler som f.eks `PATH`, `LD_LIBRARY_PATH`, `MANPATH`, `INFOPATH` og `CPPFLAGS` må utvides til å inkludere `/usr/pkg/foo`. For mer enn noen få pakker, blir denne ordningen uhåndterlig.

8.2.2.3. Symlink Style Package Management

Dette er en variant av den tidligere pakkehåndteringsteknikken. Hver pakke er installert på samme måte som den forrige ordningen. Men i stedet for å gjøre symbolkoblingen, er hver fil symlinked inn i `/usr` hierarkiet. Dette fjerner behovet for å utvide miljøvariablene. Selv om symbolkoblingene kan være opprettet av brukeren for å automatisere opprettelsen, har mange pakkeforvaltere blitt skrevet ved hjelp av denne tilnærmingen. Noen av de populære inkluderer Stow, Epkg, Graft og Depot.

Installasjonen må forfalskes, slik at pakken tror det den er installert i `/usr` skjønt i virkeligheten er den installert i `/usr/pkg` hierarkiet. Installering på denne måten er vanligvis ikke en triviell oppgave. Tenk for eksempel på at du installerer en pakke libfoo-1.1. Følgende instruksjoner kan ikke installere pakken riktig:

```
./configure --prefix=/usr/pkg/libfoo/1.1
make
make install
```

Installasjonen vil fungere, men de avhengige pakkene kan ikke kobles til libfoo som du forventer. Hvis du kompilerer en pakke som lenker mot libfoo, kan du legge merke til at den er koblet til `/usr/pkg/libfoo/1.1/lib/libfoo.so.1` i stedet for `/usr/lib/libfoo.so.1` som du forventer. Den riktige tilnærmingen er å bruke `DESTDIR` strategi for forfalske installasjon av pakken. Dette tilnærmingen fungerer som følger:

```
./configure --prefix=/usr
make
make DESTDIR=/usr/pkg/libfoo/1.1 install
```

De fleste pakker støtter denne tilnærmingen, men det er noen som ikke gjør det. For de ikke-kompatible pakkene kan det hende du må installere pakken manuelt , eller du kan finne ut at det er lettere å installere noen problematiske pakker inn i `/opt`.

8.2.2.4. Tidsstempelbasert

I denne teknikken blir en fil tidsstemplet før installasjonen av pakken. Etter installasjonen, en enkel bruk av **find** kommandoen med de riktige alternativene kan generere en logg over alle filene som er installert etter at tidsstempelfilen ble opprettet. En pakkebehandler skrevet med denne tilnærmingen er install-log.

Selv om denne ordningen har fordelen av å være enkel, har den to ulemper. Hvis filene under installasjonen er installert med et andre tidsstempel enn gjeldende tid, vil disse filene ikke spores av pakkebehandleren. Dessuten kan denne ordningen bare brukes når én pakke installeres om gangen. Loggene er ikke pålitelige hvis to pakker installeres på to forskjellige konsoller.

8.2.2.5. Sporing av installasjonsskript

I denne tilnærmingen, blir kommandoene som installasjonsskriptene utfører registrert. Det er to teknikker man kan bruke:

`LD_PRELOAD` miljøvariabelen kan settes til å peke på et bibliotek som skal forhåndslastes før installasjonen. Under installasjonen, sporer dette biblioteket pakkene som blir installert og fester seg til ulike kjørbare filer som f.eks **cp**, **install**, **mv** og sporing av systemets anrop som endrer filsystemet. For å få denne tilnærmingen til å virke, alle kjørbare filer må være dynamisk koblet uten suid- eller sgid-biten. Forhåndsinnlasting av biblioteket kan forårsake noen uønskede bivirkninger under installasjon. Derfor anbefales det at man utfører noen tester for å sørge for at pakkebehandlingen ikke bryter noe og logger alle passende filer.

Den andre teknikken er å bruke **strace**, som logger alle systemanrop som gjøres under utførelse av installasjonensskriptet.

8.2.2.6. Opprette pakkearkiver

I denne ordningen er pakkeinstallasjonen forfalsket til et separat tre som beskrevet i Symlink pakkebehandlingenen. Etter installasjon, opprettes et pakkearkiv ved hjelp av de installerte filene. Dette arkivet brukes deretter til å installere pakken enten på den lokale maskin eller kan til og med brukes til å installere pakken på andre maskiner.

Denne tilnærmingen brukes av de fleste pakkebehandlere som finnes i kommersielle distribusjoner. Eksempler på pakkeforvaltere som følger dette tilnærmingen er RPM (som for øvrig kreves av *Linux Standard Base Specification*), pkg-utils, Debian's apt, og Gentoo's Portage system. Et hint som beskriver hvordan du adopterer denne stilen av pakkehåndtering for LFS systemer ligger på *https://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt*.

Oppretting av pakkefiler som inkluderer avhengighetsinformasjon er kompleks og er utenfor omfanget av LFS.

Slackware bruker et **tar** basert system for pakke arkiv. Dette systemet håndterer med vilje ikke pakkeavhengigheter som mer komplekse pakkeforvaltere gjør. For detaljer om Slackware pakkebehandling, se *http://www.slackbook.org/html/package-management.html*.

8.2.2.7. Brukerbasert administrasjon

Denne ordningen, unik for LFS, ble utviklet av Matthias Benkmann, og er tilgjengelig fra *Hints Project*. I denne ordningen, er hver pakke installert som en separat bruker i standardplasseringer. Filer som tilhører en pakke identifiseres enkelt med å sjekke bruker-ID. Funksjonene og manglene ved denne tilnærmingen er for komplisert til å beskrive i denne delen. For detaljer, se hintene på *https://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt*.

8.2.3. Distribuere LFS på flere systemer

En av fordelene med et LFS system er at det ikke er noen filer som avhenger av plasseringen til filene på et disksystem. Kloning av et LFS bygg til en annen datamaskin med samme arkitektur som basissystemet er like enkelt som å bruke **tar** på LFS partisjonen som inneholder rotkatalogen (ca. 250 MB ukomprimert for en standard LFS bygg), kopiere den filen via nettverksoverføring eller CD-ROM til det nye systemet og utvide den. Fra det tidspunktet må noen få konfigurasjonsfiler endres. Konfigurasjonsfiler som kanskje må oppdateres inkluderer: `/etc/hosts`, `/etc/fstab`, `/etc/passwd`, `/etc/group`, `/etc/shadow`, og `/etc/ld.so.conf`.

En tilpasset kjerne må kanskje bygges for det nye systemet avhengig av forskjeller i systemmaskinvare og den originale kjerne konfigurasjonen.

Note

Det har vært noen rapporter om problemer ved kopiering mellom lignende, men ikke identiske arkitekturer. For eksempel instruksjonssettet for et Intel-system er ikke identisk med en AMD-prosessor og nyere versjoner av enkelte prosessorer kan ha instruksjoner som ikke er tilgjengelige i tidligere versjoner.

Til slutt må det nye systemet gjøres oppstartbart via Section 10.4, “Bruke GRUB til å sette opp oppstartsprosessen”.

8.3. Man-pages-5.13

Man-pages pakken inneholder over 2200 Man-sider.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 33 MB

8.3.1. Installasjon av Man-sider

Installer Man-sider ved å kjøre:

```
make prefix=/usr install
```

8.3.2. Innhold i Man-sider

Installerte filer: ulike Man-sider

Korte beskrivelser

`man pages` Beskriver C programmeringsspråksfunksjoner, viktig enhetsfiler og betydelige konfigurasjonsfiler

8.4. Iana-Etc-20220207

Iana-Etc pakken leverer data for nettverkstjenester og protokoller.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 4.7 MB

8.4.1. Installasjon av Iana-Etc

For denne pakken trenger vi bare å kopiere filene på plass:

```
cp services protocols /etc
```

8.4.2. Innhold i Iana-Etc

Installerte filer: /etc/protocols and /etc/services

Korte beskrivelser

`/etc/protocols` Beskriver de ulike DARPA Internettprotokollene som er tilgjengelig fra TCP/IP undersystemet

`/etc/services` Gir en tilordning mellom vennlige tekstnavn for internetttjenester, og deres underliggende tildelte portnumre og protokolltyper

8.5. Glibc-2.35

Glibc pakken inneholder C hovedbiblioteket. Dette biblioteket tilbyr de grunnleggende rutinene for tildeling av minne, søk i kataloger, åpne og lukke filer, lese og skrive filer, strenghåndtering, mønstertilpasning, aritmetikk og så videre.

Omtrentlig byggetid: 24 SBU
Nødvendig diskplass: 2.8 GB

8.5.1. Installasjon av Glibc

Noen av Glibc programmene bruker ikke-FHS kompatible `/var/db` mappen til å lagre kjøretidsdataene i. Bruk følgende oppdatering for å forsikre om at slike programmer lagrer kjøretidsdataene deres på de FHS-kompatible stedene:

```
patch -Np1 -i ../glibc-2.35-fhs-1.patch
```

Glibc dokumentasjonen anbefaler å bygge Glibc i en dedikert byggemappe:

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

Sørg for at **ldconfig** og **sln** verktøyene vil bli installert i `/usr/sbin`:

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

Forbered Glibc for kompilering:

```
../configure --prefix=/usr                            \
             --disable-werror                         \
             --enable-kernel=3.2                      \
             --enable-stack-protector=strong          \
             --with-headers=/usr/include              \
             libc_cv_slibdir=/usr/lib
```

Betydningen av konfigureringsalternativene:

`--disable-werror`
Dette alternativet deaktiverer alternativet -Werror sendt til GCC. Dette er nødvendig for å kjøre testpakken.

`--enable-kernel=3.2`
Dette alternativet forteller byggesystemet at denne glibc kan brukes med kjerner så gamle som 3.2. Dette betyr å generere løsninger i tilfelle et systemanrop introdusert i en senere versjon ikke kan brukes.

`--enable-stack-protector=strong`
Dette alternativet øker systemsikkerheten ved å legge til ekstra kode for å se etter bufferoverløp, for eksempel stabel (stack) knusende angrep.

`--with-headers=/usr/include`
Dette alternativet forteller byggesystemet hvor det skal finne kjernens API deklarasjoner.

`libc_cv_slibdir=/usr/lib`
Denne variabelen setter riktig bibliotek for alle systemer. Vi ønsker ikke at lib64 skal brukes.

Kompiler pakken:

```
make
```

Important

I denne delen anses testpakken for Glibc som kritisk. Ikke hopp over den under noen omstendigheter.

Vanligvis består ikke noen få tester. Testfeilene som er oppført nedenfor er vanligvis trygge å ignorere.

```
make check
```

Du kan se noen testfeil. Glibc testpakken er noe avhengig av vertssystemet. Noen få feil ut av over 4200 tester kan generelt ignoreres. Dette er en liste over de vanligste problemene som er sett for nyere versjoner av LFS:

- *io/tst-lchmod* er kjent for å mislykkes i LFS chroot miljøet.
- *misc/tst-ttyname* er kjent for å mislykkes i LFS chroot miljøet.
- *nss/tst-nss-files-hosts-multi* testen er kjent for å mislykkes hvis systemet ikke har noen ikke-tilbakekoblings IP-adresser.

Selv om det er en ufarlig melding, vil installasjonsstadiet til Glibc klage på fravær av `/etc/ld.so.conf`. Forhindre denne advarselen med:

```
touch /etc/ld.so.conf
```

Fiks Makefilen til å hoppe over en unødvendig fornuftssjekk som svikter i LFS delmiljøet:

```
sed '/test-installation/s@$(PERL)@echo not running@' -i ../Makefile
```

Installer pakken:

```
make install
```

Fiks hardkodet bane til den kjørbare lasteren i **ldd** skriptet:

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

Installer konfigurasjonsfilen og kjøretidsmappen for **nscd**:

```
cp -v ../nscd/nscd.conf /etc/nscd.conf
mkdir -pv /var/cache/nscd
```

Installer systemd støttefilene for **nscd**:

```
install -v -Dm644 ../nscd/nscd.tmpfiles /usr/lib/tmpfiles.d/nscd.conf
install -v -Dm644 ../nscd/nscd.service /usr/lib/systemd/system/nscd.service
```

Installer deretter lokalitetene som kan få systemet til å svare i en annet språk. Ingen av lokalitetene er påkrevd, men hvis noen av dem mangler, vil testpakkene til fremtidige pakker hoppe over viktig tester.

Individuelle lokaliteter kan installeres ved å bruke **localedef** programmet. For eksempel den andre **localedef** kommandoen nedenfor kombinerer `/usr/share/i18n/locales/cs_CZ` tegnsettuavhengig lokalitetsdefinisjonen med `/usr/share/i18n/charmaps/UTF-8.gz` tegnsett definisjonen og legger resultatet til `/usr/lib/locale/locale-archive` filen. Følgende instruksjoner vil installere minimumssettet med lokaliteter som er nødvendige for optimal dekning av tester:

```
mkdir -pv /usr/lib/locale
localedef -i POSIX -f UTF-8 C.UTF-8 2> /dev/null || true
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
```

Installer i tillegg lokaliteten for ditt eget land, språk og tegnsett.

Alternativt kan du installere alle lokaliteter som er oppført i `glibc-2.35/localedata/SUPPORTED` filen (den inkluderer alle lokaliteter oppført ovenfor og mange flere) samtidig med denne tidkrevende kommandoen:

```
make localedata/install-locales
```

Bruk deretter **localedef** kommandoen for å lage og installere lokaliteter som ikke er oppført i `glibc-2.35/localedata/SUPPORTED` filen når du trenger dem. For eksempel er følgende to lokaliteter nødvendig for noen tester senere i dette kapittelet:

```
localedef -i POSIX -f UTF-8 C.UTF-8 2> /dev/null || true
localedef -i ja_JP -f SHIFT_JIS ja_JP.SJIS 2> /dev/null || true
```

Note

Glibc bruker nå libidn2 når den løser internasjonalisert domenenavn. Dette er en kjøretids avhengighet. Hvis denne evnen er nødvendig, er instruksjonene for installasjon av libidn2 i *BLFS libidn2 siden*.

8.5.2. Konfigurerer Glibc

8.5.2.1. Legge til nsswitch.conf

`/etc/nsswitch.conf` filen må opprettes fordi Glibc standardene ikke fungerer bra i et nettverksmiljø.

Opprett en ny fil `/etc/nsswitch.conf` ved å kjøre følgende:

```
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. Legger til tidssonedata

Installer og sett opp tidssonedataene med følgende:

```
tar -xf ../../tzdata2021e.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
```

Betydningen av zic kommandoene:

`zic -L /dev/null ...`

Dette skaper posix tidssoner uten noen skuddsekunder. Det er konvensjonelt å legge disse i både `zoneinfo` og `zoneinfo/posix`. Det er nødvendig for å legge POSIX tidssonene i `zoneinfo`, ellers vil forskjellige testpakker rapportere feil. På et innebygd system, hvor plass er stramt og du aldri har tenkt å oppdatere tidssonene, kan du spare 1,9 MB ved å ikke bruke `posix` mappen, men noen applikasjoner eller testpakker kan produsere noen feil.

`zic -L leapseconds ...`

Dette skaper riktige tidssoner, inkludert skuddsekunder. På en innebygd system, hvor det er trangt om plass og du ikke har tenkt å oppdatere tidssonene noen gang, eller ikke bryr deg om riktig tid, kan du spar 1,9 MB ved å utelate `right` mappen.

`zic ... -p ...`

Dette oppretter `posixrules` filen. Vi bruker New York fordi POSIX krever at reglene for sommertid er i samsvar med amerikanske regler.

En måte å bestemme den lokale tidssonen på er å kjøre følgende skript:

```
tzselect
```

Etter å ha svart på noen spørsmål om lokaliteten, vil skriptet skrive ut navnet på tidssonen (f.eks. *America/Edmonton*). Det er også noen andre mulige tidssoner oppført i `/usr/share/zoneinfo` som for eksempel *Canada/Eastern* eller *EST5EDT* som ikke identifiseres av skriptet, men kan brukes.

Deretter oppretter du `/etc/localtime` filen med å kjøre:

```
ln -sfv /usr/share/zoneinfo/<xxx> /etc/localtime
```

Erstatt *<xxx>* med navnet på den valgte tidssonen (f.eks. Canada/Eastern).

8.5.2.3. Konfigurere den dynamiske lasteren

Som standard den dynamiske lasteren (`/lib/ld-linux.so.2`) søker gjennom `/lib` og `/usr/lib` for dynamiske biblioteker som trengs av programmer etter hvert som de kjøres. Men hvis det er biblioteker i andre kataloger enn `/lib` og `/usr/lib`, disse må legges til i `/etc/ld.so.conf` filen for at den dynamiske lasteren skal finne dem. To kataloger som er allment kjent å inneholde flere biblioteker er `/usr/local/lib` og `/opt/lib`, så legg disse mappene til den dynamiske lasterens søkebane.

Opprett en ny fil `/etc/ld.so.conf` ved å kjøre følgende:

```
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf
/usr/local/lib
/opt/lib

EOF
```

Om ønskelig kan den dynamiske lasteren også søke i en mappe og inkludere innholdet i filene som finnes der. Vanligvis vil filene i denne mappen inkludere en linje som spesifiserer ønsket biblioteksti. For å legge til denne funksjonen, kjør følgende kommandoer:

```
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. Innhold i Glibc

Installerte programmer: gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, ld.so (symlink to ld-linux-x86-64.so.2 or ld-linux.so.2), locale, localedef, makedb, mtrace, nscd, pcprofiledump, pldd, sln, sotruss, sprof, tzselect, xtrace, zdump, og zic

Installerte biblioteker: 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, libcrypt.{a,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, og libutil.{a,so.1}

Installerte mapper: /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, /var/cache/nscd, og /var/lib/nss_db

Korte beskrivelser

gencat Genererer meldingskataloger

getconf Viser systemkonfigurasjonsverdiene for filsystem spesifikke variabler

getent Henter oppføringer fra en administrativ database

iconv	Utfører tegnsettkonvertering
iconvconfig	Skaper hurtiglastings **iconv** modulkonfigurasjons filer
ldconfig	Konfigurerer dynamiske lenker til kjøretidsbindingene
ldd	Rapporter hvilke delte biblioteker som kreves av hvert gitt program eller delte bibliotek
lddlibc4	Assisterer **ldd** med objektfiler. Det eksisterer ikke på nyere arkitekturer som x86_64
locale	Skriver ut forskjellig informasjon om gjeldende lokalitet
localedef	Kompilerer lokalitets spesifikasjoner
makedb	Oppretter en enkel database fra tekst inndata
mtrace	Leser og tolker en minnesporingsfil og viser et sammendrag i menneskelestbart format
nscd	En nisse (daemon) som gir et hurtiglager for de vanligste navne tjenesteforespørsler
pcprofiledump	Dumper informasjon generert av PC profiling
pldd	Viser dynamiske delte objekter som brukes av kjørende prosesser
sln	En statisk koblet **ln** programm
sotruss	Sporer delte biblioteksprosedyrekall for en spesifisert kommando
sprof	Leser og viser profileringsdata for delte objekter
tzselect	Spør brukeren om lokaliteten til systemet og rapporterer den tilsvarende tidssonebeskrivelsen
xtrace	Sporer kjøringen av et program ved å skrive ut gjeldende utført funksjon
zdump	Tidssone dumperen
zic	Tidssonekompilatoren
`ld-*.so`	Hjelpeprogrammet for kjørbare delte biblioteker
`libBrokenLocale`	Brukes internt av Glibc som et grovt hack for å få ødelagte programmer (f.eks. noen Motif-applikasjoner) kjørende. Se kommentarer i `glibc-2.35/locale/broken_cur_max.c` for mer informasjon
`libanl`	Et asynkront navneoppslagsbibliotek
`libc`	C hovedbiblioteket
`libc_malloc_debug`	Slår på minneallokeringskontroll når den er forhåndslastet
`libcrypt`	Kryptografibiblioteket
`libdl`	Dummy bibliotek som ikke inneholder noen funksjoner. Tidligere var den dynamisk koblingsgrensesnittbibliotek, funksjonene er nå i `libc`
`libg`	Dummy bibliotek som ikke inneholder noen funksjoner. Tidligere var det et kjøretidsbibliotek for **g++**
`libm`	Det matematiske biblioteket
`libmvec`	Matematisk vektorbibliotek, koblet inn etter behov når `libm` blir brukt
`libmcheck`	Slår på minneallokeringskontroll når den er koblet til
`libmemusage`	Brukt av **memusage** for å hjelpe til med å samle inn informasjon om minnebruken til et program

`libnsl`	Nettverkstjenestebiblioteket, nå avviklet
`libnss_*`	Navnetjenestebrytermodulene, som inneholder funksjoner for å løse vertsnavn, brukernavn, gruppenavn, aliaser, tjenester, protokoller osv. Lastet av `libc` ifølge konfigurasjon i `/etc/nsswitch.conf`
`libpcprofile`	Kan forhåndslastes til PC profile en kjørbar fil
`libpthread`	Dummy bibliotek som ikke inneholder noen funksjoner. Tidligere inneholdt den funksjoner som gir de fleste grensesnittene som er spesifisert av POSIX.1b Realtime Extension, nå er funksjonene i `libc`
`libresolv`	Inneholder funksjoner for å lage, sende og tolke pakker til domenenavnservere
`librt`	Inneholder funksjoner som gir de fleste grensesnittene som er spesifisert av POSIX.1b Realtime Extension
`libthread_db`	Inneholder funksjoner som er nyttige for å bygge feilsøkere for flertrådede programmer
`libutil`	Dummy bibliotek som ikke inneholder noen funksjoner. Tidligere inneholdt den kode for “standard” funksjoner som brukes i mange forskjellige Unix verktøy. Disse funksjonene er nå i `libc`

8.6. Zlib-1.2.11

Zlib pakken inneholder komprimerings- og dekompresjonsrutiner som brukes av noen programmer.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 5.0 MB

8.6.1. Installasjon av Zlib

Forbered Zlib for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

Fjern et ubrukelig statisk bibliotek:

```
rm -fv /usr/lib/libz.a
```

8.6.2. Innhold i Zlib

Installerte biblioteker: libz.so

Korte beskrivelser

`libz` Inneholder komprimerings- og dekompresjonsfunksjoner som brukes av noen programmer

8.7. Bzip2-1.0.8

Bzip2 pakken inneholder programmer for komprimering og dekomprimering av filer. Komprimering av tekstfiler med **bzip2** gir mye bedre kompresjonsprosent enn med den tradisjonelle **gzip**.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 7.2 MB

8.7.1. Installasjon av Bzip2

Bruk en oppdatering som vil installere dokumentasjonen for denne pakken:

```
patch -Np1 -i ../bzip2-1.0.8-install_docs-1.patch
```

Følgende kommando sikrer at installasjonen av symbolske lenker er relative:

```
sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
```

Sørg for at man sidene er installert på riktig sted:

```
sed -i "s@(PREFIX)/man@(PREFIX)/share/man@g" Makefile
```

Forbered Bzip2 for kompilering med:

```
make -f Makefile-libbz2_so
make clean
```

Betydningen av make parameteren:

`-f Makefile-libbz2_so`
Vil føre til at Bzip2 bygges med en annen `Makefile` fil, i dette tilfellet `Makefile-libbz2_so` filen, som skaper en dynamisk `libbz2.so` bibliotek og lenker Bzip2 verktøyene mot det.

Kompiler og test pakken:

```
make
```

Installer programmene:

```
make PREFIX=/usr install
```

Installer det delte biblioteket:

```
cp -av libbz2.so.* /usr/lib
ln -sv libbz2.so.1.0.8 /usr/lib/libbz2.so
```

Installer den delte **bzip2** binær inn i `/usr/bin` mappen, og erstatt to eksemplarer av **bzip2** med symbolske lenker:

```
cp -v bzip2-shared /usr/bin/bzip2
for i in /usr/bin/{bzcat,bunzip2}; do
  ln -sfv bzip2 $i
done
```

Fjern et ubrukelig statisk bibliotek:

```
rm -fv /usr/lib/libbz2.a
```

8.7.2. Innhold i Bzip2

Installerte programmer: bunzip2 (linker til bzip2), bzcat (linker til bzip2), bzcmp (linker til bzdiff), bzdiff, bzegrep (linker til bzgrep), bzfgrep (linker til bzgrep), bzgrep, bzip2, bzip2recover, bzless (linker til bzmore), og bzmore
Installerte biblioteker: libbz2.so
Installert mappe: /usr/share/doc/bzip2-1.0.8

Korte beskrivelser

bunzip2 Dekomprimerer bzippede filer

bzcat Dekomprimerer til standard utgang

bzcmp Kjører **cmp** på bzippede filer

bzdiff Kjører **diff** på bzippede filer

bzegrep Kjører **egrep** på bzippede filer

bzfgrep Kjører **fgrep** på bzippede filer

bzgrep Kjører **grep** på bzippede filer

bzip2 Komprimerer filer ved å bruke Burrows-Wheeler sortering på blokktekst komprimeringsalgoritme med Huffman-koding; kompresjonshastigheten er bedre enn det som oppnås med mer konvensjonelle kompressorer som bruker “Lempel-Ziv” algoritmer, som **gzip**

bzip2recover Prøver å gjenopprette data fra skadde bzippede filer

bzless Kjører **less** på bzippede filer

bzmore Kjører **more** på bzippede filer

`libbz2` Biblioteket implementerer tapsfri, blokksorterende data komprimering ved å bruke Burrows-Wheeler-algoritmen

8.8. Xz-5.2.5

Xz pakken inneholder programmer for komprimering og dekomprimering av filer. Det gir muligheter for lzma og den nyere xz komprimerings formatene. Komprimering av tekstfiler med **xz** gir en bedre kompresjonsprosent enn med de tradisjonelle **gzip** eller **bzip2** kommandoene.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 15 MB

8.8.1. Installasjon av Xz

Forbered Xz for kompilering med:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.2.5
```

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.8.2. Innhold i Xz

Installerte programmer: lzcat (lenker til xz), lzcmp (lenker til xzdiff), lzdiff (lenker til xzdiff), lzegrep (lenker til xzgrep), lzfgrep (lenker til xzgrep), lzgrep (lenker til xzgrep), lzless (lenker til xzless), lzma (lenker til xz), lzmadec, lzmainfo, lzmore (lenker til xzmore), unlzma (lenker til xz), unxz (lenker til xz), xz, xzcat (lenker til xz), xzcmp (lenker til xzdiff), xzdec, xzdiff, xzegrep (lenker til xzgrep), xzfgrep (lenker til xzgrep), xzgrep, xzless, og xzmore
Installerte biblioteker: liblzma.so
Installerte mapper: /usr/include/lzma og /usr/share/doc/xz-5.2.5

Korte beskrivelser

lzcat Dekomprimerer til standard utgang

lzcmp Kjører **cmp** på LZMA komprimerte filer

lzdiff Kjører **diff** på LZMA komprimerte filer

lzegrep Kjører **egrep** på LZMA komprimerte filer

lzfgrep Kjører **fgrep** på LZMA komprimerte filer

lzgrep Kjører **grep** på LZMA komprimerte filer

lzless Kjører **less** på LZMA komprimerte filer

lzma Komprimerer eller dekomprimerer filer ved å bruke LZMA formatet

lzmadec En liten og rask dekoder for LZMA komprimerte filer

lzmainfo Viser informasjon som er lagret i den komprimerte LZMA filoverskriften

lzmore Kjører **more** på LZMA komprimerte filer

unlzma Dekomprimerer filer ved å bruke LZMA formatet

unxz Dekomprimerer filer ved å bruke XZ formatet

xz Komprimerer eller dekomprimerer filer ved å bruke XZ formatet

xzcat Dekomprimerer til standard utgang

xzcmp Kjører **cmp** på XZ komprimerte filer

xzdec En liten og rask dekoder for XZ komprimerte filer

xzdiff Kjører **diff** på XZ komprimerte filer

xzegrep Kjører **egrep** på XZ komprimerte filer

xzfgrep Kjører **fgrep** på XZ komprimerte filer

xzgrep Kjører **grep** på XZ komprimerte filer

xzless Kjører **less** på XZ komprimerte filer

xzmore Kjører **more** på XZ komprimerte filer

`liblzma` Bibliotek som implementerer tapsfri, blokksorterende data komprimering ved å bruke Lempel-Ziv-Markov kjedealgoritmen

8.9. Zstd-1.5.2

Zstandard er en sanntidskomprimeringsalgoritme som gir høyt kompresjonsforhold. Den tilbyr et veldig bredt spekter av kompresjons/hastighets avveininger, samtidig som den støttes av en veldig rask dekoder.

Omtrentlig byggetid: 1.1 SBU
Nødvendig diskplass: 55 MB

8.9.1. Installasjon av Zstd

Kompiler pakken:

```
make
```

Note

I testutdataen er det flere steder som angir 'failed'. Disse er forventet og bare 'FAIL' er en reell testfeil. Det skal ikke være noen testfeil.

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make prefix=/usr install
```

Fjern det statiske biblioteket:

```
rm -v /usr/lib/libzstd.a
```

8.9.2. Innhold i Zstd

Installerte programmer: zstd, zstdcat (lenker til zstd), zstdgrep, zstdless, zstdmt (lenker til zstd), og unzstd (lenker til zstd)
Installert bibliotek: libzstd.so

Korte beskrivelser

zstd Komprimerer eller dekomprimerer filer ved å bruke ZSTD formatet

zstdgrep Kjører **grep** på ZSTD komprimerte filer

zstdless Kjører **less** på ZSTD komprimerte filer

`libzstd` Biblioteket implementerer tapsfri data komprimering ved å bruke ZSTD algoritmen

8.10. File-5.41

File pakken inneholder et verktøy for å bestemme typen av en gitt fil eller filer.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 15 MB

8.10.1. Installasjon av File

Forbered File for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.10.2. Innholdet i File

Installerte programmer: file
Installert bibliotek: libmagic.so

Korte beskrivelser

file Prøver å klassifisere hver gitt fil; den gjør dette ved å utføre flere tester—filsystemtester, magiske talltester og språk tester

`libmagic` Inneholder rutiner for magisk tallgjenkjenning, brukt av **file** programmet

8.11. Readline-8.1.2

Readline pakken er et sett med biblioteker som tilbyr kommandolinje redigerings- og historikkfunksjoner.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 15 MB

8.11.1. Installasjon av Readline

Å installere Readline på nytt vil føre til at de gamle bibliotekene flyttes til <libraryname>.old. Selv om dette normalt ikke er et problem, i noen tilfeller kan det utløse en koblingsfeil i **ldconfig**. Dette kan være unngås ved å utstede følgende to seds:

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

Forbered Readline for kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --with-curses    \
            --docdir=/usr/share/doc/readline-8.1.2
```

Betydningen av konfigureringsalternativet:

`--with-curses`
Dette alternativet forteller Readline at det kan finne termcap bibliotekfunksjoner i curses biblioteket, i stedet for et separat termcap bibliotek. Det gjør det mulig å generere en korrekt `readline.pc` fil.

Kompiler pakken:

```
make SHLIB_LIBS="-lncursesw"
```

Betydningen av make alternativet:

`SHLIB_LIBS="-lncursesw"`
Dette alternativet tvinger Readline til å lenke mot `libncursesw` biblioteket.

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make SHLIB_LIBS="-lncursesw" install
```

Hvis ønskelig, installer dokumentasjonen:

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.1.2
```

8.11.2. Innhold i Readline

Installerte biblioteker: libhistory.so og libreadline.so
Installerte mapper: /usr/include/readline og /usr/share/doc/readline-8.1.2

Korte beskrivelser

`libhistory` Gir et konsistent brukergrensesnitt for tilbakekalling av linjer fra historien

`libreadline` Gir et sett med kommandoer for å manipulere tekst som er skrevet inn i et interaktiv økt av et program

8.12. M4-1.4.19

M4 pakken inneholder en makroprosessor.

Omtrentlig byggetid: 0.7 SBU
Nødvendig diskplass: 49 MB

8.12.1. Installasjon av M4

Forbered M4 for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.12.2. Innhold i M4

Installert program: m4

Korte beskrivelser

m4 Kopierer de gitte filene mens de utvider makroene som de inneholder. Disse makroene er enten innebygde eller brukerdefinerte og kan ta et hvilket som helst antall argumenter. Foruten å utføre makroutvidelse, **m4** har innebygde funksjoner for å inkludere navngitte filer, kjører Unix kommandoer, utfører heltallsaritmetikk, manipulere tekst, rekursjon osv. **m4** programmet kan brukes enten som en grenseflate til en kompilator eller som en makroprosessor på egen hånd

8.13. Bc-5.2.2

Bc pakken inneholder et vilkårlig behandlingsspråk for numerisk presisjon.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 7.1 MB

8.13.1. Installasjon av Bc

Forbered Bc for kompilering:

```
CC=gcc ./configure --prefix=/usr -G -O3
```

Betydningen av konfigureringsalternativene:

`CC=gcc`
Denne parameteren spesifiserer kompilatoren som skal brukes.

`-O3`
Spesifiser optimaliseringen som skal brukes.

`-G`
Utelater deler av testpakken som ikke vil fungere uten GNU bc tilstede.

Kompiler pakken:

```
make
```

For å teste bc, kjør:

```
make test
```

Installer pakken:

```
make install
```

8.13.2. Innholdet i Bc

Installerte programmer: bc og dc

Korte beskrivelser

bc En kommandolinjekalkulator

dc En omvendt-polert kommandolinjekalkulator

8.14. Flex-2.6.4

Flex pakken inneholder et verktøy for å generere programmerer som gjenkjenner mønstre i tekst.

Omtrentlig byggetid: 0.4 SBU
Nødvendig diskplass: 32 MB

8.14.1. Installasjon av Flex

Forbered Flex for kompilering:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/flex-2.6.4 \
            --disable-static
```

Kompiler pakken:

```
make
```

For å teste resultatene (ca. 0,5 SBU), utsted:

```
make check
```

Installer pakken:

```
make install
```

Noen få programmer vet ikke om **flex** ennå og prøver å kjøre forgjengeren, **lex**. For å støtte de programmene, opprett en symbolsk lenke kalt `lex` som kjører `flex` i **lex** emuleringsmodus:

```
ln -sv flex /usr/bin/lex
```

8.14.2. Innhold i Flex

Installerte programmer: flex, flex++ (lenker til flex), og lex (lenker til flex)
Installerte biblioteker: libfl.so
Installert mappe: /usr/share/doc/flex-2.6.4

Korte beskrivelser

flex Et verktøy for å generere programmer som gjenkjenner mønstre i tekst; det gir mulighet for allsidigheten til å spesifisere reglene for mønstersøking, eliminere behovet for å utvikle et spesialisert program

flex++ En utvidelse av flex brukes til å generere C++ kode og klasser. Det er en symbolsk kobling til **flex**

lex En symbolsk lenke som kjører **flex** i **lex** emuleringsmodus

`libfl` `flex` biblioteket

8.15. Tcl-8.6.12

Tcl pakken inneholder Tool Command Language, et robust skriptspråk for generelt bruk. Expect pakken er skrevet i Tcl språket.

Omtrentlig byggetid: 3.4 SBU
Nødvendig diskplass: 87 MB

8.15.1. Installasjon av Tcl

Denne pakken og de to neste (Expect og DejaGNU) er installert for å støtte kjøring av testpakkene for binutils og GCC og andre pakker. Å installere tre pakker for testformål kan virke overdrevent, men det er veldig betryggende, om ikke avgjørende, å vite at de viktigste verktøyene fungerer som de skal.

Pakk først ut dokumentasjonen ved å gi følgende kommando:

```
tar -xf ../tcl8.6.12-html.tar.gz --strip-components=1
```

Forbered Tcl for kompilering:

```
SRCDIR=$(pwd)
cd unix
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            $([ "$(uname -m)" = x86_64 ] && echo --enable-64bit)
```

Betydningen av konfigureringsalternativene:

`$([ "$(uname -m)" = x86_64 ] && echo --enable-64bit)`
construct *`$(<shell command>)`* erstattes av utdataene fra shell kommandoen. Her er denne utgangen tom hvis den kjøres på en 32-bits maskin, og er *`--enable-64bit`* hvis du kjører på en 64-bits maskin.

Bygg pakken:

```
make

sed -e "s|$SRCDIR/unix|/usr/lib|" \
    -e "s|$SRCDIR|/usr/include|"  \
    -i tclConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/tdbc1.1.3|/usr/lib/tdbc1.1.3|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.3/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/tdbc1.1.3/library|/usr/lib/tcl8.6|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.3|/usr/include|"            \
    -i pkgs/tdbc1.1.3/tdbcConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/itcl4.2.2|/usr/lib/itcl4.2.2|" \
    -e "s|$SRCDIR/pkgs/itcl4.2.2/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/itcl4.2.2|/usr/include|"            \
    -i pkgs/itcl4.2.2/itclConfig.sh

unset SRCDIR
```

De ulike “sed” instruksjonene etter “make” kommandoen fjerner referanser til byggemappen fra konfigurasjonsfilene og erstatter dem med installasjonsmappen. Dette er ikke obligatorisk for resten av LFS, men kan være nødvendig i tilfelle en pakke bygget senere bruker Tcl.

For å teste resultatene, utsted:

```
make test
```

Installer pakken:

```
make install
```

Gjør det installerte biblioteket skrivbart slik at feilsøkingssymboler kan fjernes senere:

```
chmod -v u+w /usr/lib/libtcl8.6.so
```

Installer Tcl sine deklarasjoner. Den neste pakken, Expect, krever dem.

```
make install-private-headers
```

Lag nå en nødvendig symbolsk kobling:

```
ln -sfv tclsh8.6 /usr/bin/tclsh
```

Gi nytt navn til en man side som er i konflikt med en man side for Perl:

```
mv /usr/share/man/man3/{Thread,Tcl_Thread}.3
```

Hvis du lastet ned den valgfrie dokumentasjonen, installer den ved å utstede den følgende kommandoer:

```
mkdir -v -p /usr/share/doc/tcl-8.6.12
cp -v -r  ../html/* /usr/share/doc/tcl-8.6.12
```

8.15.2. Innhold i Tcl

Installerte programmer: tclsh (link to tclsh8.6) og tclsh8.6
Installert bibliotek: libtcl8.6.so og libtclstub8.6.a

Korte beskrivelser

tclsh8.6	Tcl kommandoskallet
tclsh	En lenke til tclsh8.6
`libtcl8.6.so`	Tcl biblioteket
`libtclstub8.6.a`	Tcl Stub biblioteket

8.16. Expect-5.45.4

Expect pakken inneholder verktøy for å automatisere, via skriptede dialoger, interaktive applikasjoner som f.eks **telnet**, **ftp**, **passwd**, **fsck**, **rlogin**, og **tip**. Expect er også nyttig for å teste disse samme applikasjoner i tillegg til å lette alle slags oppgaver som er uoverkommelige vanskelig med noe annet. DejaGnu rammeverket er skrevet i Expect.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 3.9 MB

8.16.1. Installasjon av Expect

Forbered Expect for kompilering:

```
./configure --prefix=/usr           \
            --with-tcl=/usr/lib     \
            --enable-shared         \
            --mandir=/usr/share/man \
            --with-tclinclude=/usr/include
```

Betydningen av konfigureringsalternativene:

`--with-tcl=/usr/lib`
Denne parameteren er nødvendig for å fortelle **configure** hvor **tclConfig.sh** skriptet er plassert.

`--with-tclinclude=/usr/include`
Dette forteller Expect eksplisitt hvor du finner Tcls interne deklarasjoner.

Bygg pakken:

```
make
```

For å teste resultatene, utsted:

```
make test
```

Installer pakken:

```
make install
ln -svf expect5.45.4/libexpect5.45.4.so /usr/lib
```

8.16.2. Innhold i Expect

Installert program: expect
Installert bibliotek: libexpect5.45.4.so

Korte beskrivelser

expect	Kommuniserer med andre interaktive programmer iht til et skript
`libexpect-5.45.4.so`	Inneholder funksjoner som gjør at Expect kan brukes som en Tcl utvidelse eller brukes direkte fra C eller C++ (uten Tcl)

8.17. DejaGNU-1.6.3

DejaGnu pakken inneholder et rammeverk for å kjøre test pakker på GNU-verktøy. Det er skrevet i **expect**, som selv bruker Tcl verktøykommandospråk (Tool Command Language).

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 6.9 MB

8.17.1. Installasjon av DejaGNU

Upstream anbefaler å bygge DejaGNU i en dedikert byggemappe :

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

Forbered DejaGNU for kompilering:

```
../configure --prefix=/usr
makeinfo --html --no-split -o doc/dejagnu.html ../doc/dejagnu.texi
makeinfo --plaintext       -o doc/dejagnu.txt  ../doc/dejagnu.texi
```

Bygg og installer pakken:

```
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
```

For å teste resultatene, utsted:

```
make check
```

8.17.2. Innhold i DejaGNU

Installert program: dejagnu og runtest

Korte beskrivelser

dejagnu DejaGNU hjelpekommandostarter

runtest Et innpakningsskript som lokaliserer riktig **expect** skall og kjører deretter DejaGNU

8.18. Binutils-2.38

Binutils pakken inneholder en linker, en assembler og annet verktøy for håndtering av objektfiler.

Omtrentlig byggetid: 6.1 SBU
Nødvendig diskplass: 4.6 GB

8.18.1. Installasjon av Binutils

Kontroller at PTY-ene fungerer som de skal inne i chroot miljøet ved å utføre en enkel test:

```
expect -c "spawn ls"
```

Denne kommandoen skal gi følgende utdata:

```
spawn ls
```

Hvis utdataene i stedet inkluderer meldingen nedenfor, er miljøet ikke satt opp for riktig PTY drift. Dette problemet må løses før testpakkene for Binutils og GCC kjøres:

```
The system has no more ptys.
Ask your system administrator to create more.
```

Upstream har laget en oppdatering for å fikse et problem ved bruk av binutils for å bygge noen BLFS pakker med Optimalisering av koblingstid (Link Time Optimization) (LTO) aktivert. Bruk det nå:

```
patch -Np1 -i ../binutils-2.38-lto_fix-1.patch
```

Lag nå en rettelse identifisert upstream som påvirker byggingen av noen pakker:

```
sed -e '/R_386_TLS_LE /i \   || (TYPE) == R_386_TLS_IE \\' \
    -i ./bfd/elfxx-x86.h
```

Binutils dokumentasjonen anbefaler å bygge Binutils i en dedikert byggemappe:

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

Forbered Binutils for kompilering:

```
../configure --prefix=/usr       \
             --enable-gold       \
             --enable-ld=default \
             --enable-plugins    \
             --enable-shared     \
             --disable-werror    \
             --enable-64-bit-bfd \
             --with-system-zlib
```

Betydningen av konfigureringsparametrene:

`--enable-gold`
Bygg gold linken og installer den som ld.gold (ved siden av standard linker).

`--enable-ld=default`
Bygg den originale bfd linkeren og installer den som både ld (som er standard linker) og ld.bfd.

`--enable-plugins`
Aktiverer støtte for programtillegg for linkeren.

`--enable-64-bit-bfd`
Aktiverer 64 bits støtte (på verter med smalere ordstørrelser). Kanskje ikke nødvendig på 64 bits systemer, men skader ikke.

`--with-system-zlib`
Bruk det installerte zlib biblioteket i stedet for å bygge den inkluderte versjonen.

Kompiler pakken:

```
make tooldir=/usr
```

Betydningen av make parameteren:

`tooldir=/usr`
Vanligvis er verktøykatalogen (mappen der de kjørbare filene vil til slutt bli lokalisert i) satt til `$(exec_prefix)/$(target_alias)`. For eksempel, x86_64-maskiner vil utvide det til `/usr/x86_64-pc-linux-gnu`. Fordi dette er et tilpasset system, er denne målspesifikke katalogen i `/usr` ikke påkrevd. `$(exec_prefix)/$(target_alias)` ville vært brukt hvis systemet ble brukt til å krysskompilere (for eksempel kompilering av en pakke på en Intel maskin som genererer kode som kan kjøres på PowerPC maskiner).

Important

Testpakken for Binutils i denne delen anses som kritisk. Ikke hopp over det under noen omstendigheter.

Test resultatene:

```
make -k check
```

Installer pakken:

```
make tooldir=/usr install
```

Fjern ubrukelige statiske biblioteker:

```
rm -fv /usr/lib/lib{bfd,ctf,ctf-nobfd,opcodes}.a
```

8.18.2. Innhold i Binutils

Installerte programmer:	addr2line, ar, as, c++filt, dwp, elfedit, gprof, ld, ld.bfd, ld.gold, nm, objcopy, objdump, ranlib, readelf, size, strings, og strip
Installerte biblioteker:	libbfd.so, libctf.so, libctf-nobfd.so, og libopcodes.so
Installert mappe:	/usr/lib/ldscripts

Korte beskrivelser

addr2line	Oversetter programadresser til filnavn og linjenumre; gitt en adresse og navnet på en kjørbar fil, bruker den feilsøkings informasjonen i den kjørbare filen for å bestemme hvilken kildefil og linje nummer som er knyttet til adressen

ar	Oppretter, endrer og trekker ut fra arkiver
as	En assembler som setter sammen utdataene til **gcc** inn i objektfiler
c++filt	Brukes av linkeren til å ikke ødelegge C++ og Java symboler og hindre overbelastede funksjoner å krasje
dwp	DWARF pakkeverktøyet
elfedit	Oppdaterer ELF deklarasjonen til ELF filer
gprof	Viser profildata for kallgrafen
ld	En linker som kombinerer en rekke objekt og arkivfiler inn i en enkelt fil, flytter dataene deres og rydder opp i symbolreferanser
ld.gold	En nedskalert versjon av ld som bare støtter objektfil formatet elf
ld.bfd	Hardlenke til **ld**
nm	Viser symbolene som forekommer i en gitt objektfil
objcopy	Oversetter én type objektfil til en annen
objdump	Viser informasjon om den gitte objektfilen, med alternativer kontrollerer den hvilken informasjonen som skal vises; informasjonen som vises er nyttig for programmerere som jobber med kompileringens verktøy
ranlib	Genererer en indeks over innholdet i et arkiv og lagrer det i arkivet; indeksen viser alle symbolene som er definert av arkivmedlemmer som er flyttbare objektfiler
readelf	Viser informasjon om binærfiler av ELF typen
size	Viser seksjonsstørrelsene og totalstørrelsen for de gitte objektfilene
strings	Utdata, for hver gitt fil, sekvensene av utskrivbare tegn som er av minst den angitte lengden (som standard til fire); for objektfiler skriver den som standard bare strengene fra initialiserings- og lastingsseksjonene mens for andre typer filer, skanner den hele filen
strip	Kaster symboler fra objektfiler
`libbfd`	Biblioteket med binære filbeskrivelser
`libctf`	Compat ANSI-C Type Format støttebibliotek for feilsøking
`libctf-nobfd`	En libctf variant som ikke bruker libbfd funksjonalitet
`libopcodes`	Et bibliotek for å håndtere opkoder— “leselig tekst” versjoner av instruksjoner for prosessoren; den brukes til å bygge verktøy som **objdump**

8.19. GMP-6.2.1

GMP pakken inneholder matematikkbiblioteker. Disse har nyttige funksjoner for vilkårlig presisjonsaritmetikk.

Omtrentlig byggetid: 1.0 SBU
Nødvendig diskplass: 52 MB

8.19.1. Installasjon av GMP

Note

Hvis du bygger for 32-bit x86, men du har en CPU som er i stand til å kjøre 64-bits kode *og* du har spesifisert `CFLAGS` i miljøet vil konfigureringsskriptet forsøk å konfigurere for 64-biter og mislykkes. Unngå dette ved å påkalle configure kommandoen nedenfor med

```
ABI=32 ./configure ...
```

Note

Standardinnstillingene til GMP produserer biblioteker optimalisert for vertsprosessoren. Hvis det er ønskelig med biblioteker egnet for prosessorer mindre kapable enn vertens CPU, kan generiske biblioteker bli opprettet ved å kjøre følgende:

```
cp -v configfsf.guess config.guess
cp -v configfsf.sub   config.sub
```

Forbered GMP for kompilering:

```
./configure --prefix=/usr    \
            --enable-cxx     \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.2.1
```

Betydningen av de nye konfigureringsalternativene:

`--enable-cxx`
Denne parameteren aktiverer C++ støtte

`--docdir=/usr/share/doc/gmp-6.2.1`
Denne variabelen spesifiserer riktig sted for dokumentasjon.

Kompiler pakken og generer HTML dokumentasjonen:

```
make
make html
```

Important

Testpakken for GMP i denne delen anses som kritisk. Ikke hopp over det under noen omstendigheter.

Test resultatene:

```
make check 2>&1 | tee gmp-check-log
```

Caution

Koden i gmp er svært optimalisert for prosessoren hvor den er bygget. Noen ganger vil koden som oppdager prosessoren feilidentifisere systemets evner og det vil være feil i testene eller andre applikasjoner som bruker gmp bibliotekene med meldingen "Illegal instruction". I dette tilfellet bør gmp rekonfigureres med alternativet --build=x86_64-pc-linux-gnu og gjenoppbygges.

Sørg for at alle 197 testene i testpakken besto. Sjekk resultatene ved å gi følgende kommando:

```
awk '/# PASS:/{total+=$3} ; END{print total}' gmp-check-log
```

Installer pakken og dens dokumentasjon:

```
make install
make install-html
```

8.19.2. Innhold i GMP

Installerte biblioteker: libgmp.so og libgmpxx.so
Installert mappe: /usr/share/doc/gmp-6.2.1

Korte beskrivelser

`libgmp` Inneholder matematiske presisjonsfunksjoner

`libgmpxx` Inneholder C++ matematiske presisjonsfunksjoner

8.20. MPFR-4.1.0

MPFR-pakken inneholder matematiske funksjoner med flere presisjoner.

Omtrentlig byggetid: 0.8 SBU
Nødvendig diskplass: 38 MB

8.20.1. Installasjon av MPFR

Forbered MPFR for kompilering:

```
./configure --prefix=/usr        \
            --disable-static     \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.1.0
```

Kompiler pakken og generer HTML dokumentasjonen:

```
make
make html
```

Important

Testpakken for MPFR i denne delen anses som kritisk. Ikke hopp over den under noen omstendigheter.

Test resultatene og sørg for at alle tester består:

```
make check
```

Installer pakken og dens dokumentasjon:

```
make install
make install-html
```

8.20.2. Innhold i MPFR

Installerte biblioteker: libmpfr.so
Installert mappe: /usr/share/doc/mpfr-4.1.0

Korte beskrivelser

`libmpfr` Inneholder matematiske funksjoner med flere presisjoner

8.21. MPC-1.2.1

MPC pakken inneholder et bibliotek for aritmetikk av komplekse tall med vilkårlig høy presisjon og korrekt avrunding av resultatet.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 21 MB

8.21.1. Installasjon av MPC

Forbered MPC for kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.2.1
```

Kompiler pakken og generer HTML dokumentasjonen:

```
make
make html
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken og dens dokumentasjon:

```
make install
make install-html
```

8.21.2. Innhold i MPC

Installerte biblioteker: libmpc.so
Installert mappe: /usr/share/doc/mpc-1.2.1

Korte beskrivelser

`libmpc` Inneholder komplekse matematiske funksjoner

8.22. Attr-2.5.1

Attr pakken inneholder verktøy for å administrere den utvidede attributter på filsystemobjekter.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 4.1 MB

8.22.1. Installasjon av Attr

Forbered Attr for kompilering:

```
./configure --prefix=/usr     \
            --disable-static  \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.5.1
```

Kompiler pakken:

```
make
```

Testene må kjøres på et filsystem som støtter utvidete attributter som filsystemene ext2, ext3 eller ext4. For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.22.2. Innhold i Attr

Installerte programmer: attr, getfattr, and setfattr
Installert bibliotek: libattr.so
Installerte mapper: /usr/include/attr og /usr/share/doc/attr-2.5.1

Korte beskrivelser

attr Utvider attributter på filsystemobjekter

getfattr Henter de utvidede attributtene til filsystemobjekter

setfattr Angir de utvidede attributtene til filsystemobjekter

`libattr` Inneholder bibliotekfunksjonene for å manipulere utvidede attributter

8.23. Acl-2.3.1

Acl pakken inneholder verktøy for å administrere tilgangskontrolllister, som brukes til å definere mer finmaskede skjønnsmessige tilgangsrettigheter for filer og mapper.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 6.1 MB

8.23.1. Installasjon av Acl

Forbered Acl for kompilering:

```
./configure --prefix=/usr         \
            --disable-static      \
            --docdir=/usr/share/doc/acl-2.3.1
```

Kompiler pakken:

```
make
```

Acl testene må kjøres på et filsystem som støtter tilgangs kontroller etter Coreutils er bygget med Acl biblioteker. Hvis ønskelig, gå tilbake til denne pakken og kjør **make check** etter at Coreutils har blitt bygget senere i dette kapittelet.

Installer pakken:

```
make install
```

8.23.2. Innhold i Acl

Installerte programmer: chacl, getfacl, og setfacl
Installert bibliotek: libacl.so
Installerte mapper: /usr/include/acl og /usr/share/doc/acl-2.3.1

Korte beskrivelser

chacl Endrer tilgangskontrolllisten til en fil eller mappe

getfacl Henter filtilgangskontrolllister

setfacl Angir filtilgangskontrolllister

`libacl` Inneholder bibliotekfunksjonene for å manipulere tilgangskontrolllister

8.24. Libcap-2.63

Libcap pakken implementerer brukerromsgrensesnittene til POSIX 1003.1e funksjoner tilgjengelig i Linux kjerner. Disse egenskapene er en partisjonering av det allmektige root privilegiet i et sett med distinkte privilegier.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 2.7 MB

8.24.1. Installasjon av Libcap

Hindre at statiske biblioteker blir installert:

```
sed -i '/install -m.*STA/d' libcap/Makefile
```

Kompiler pakken:

```
make prefix=/usr lib=lib
```

Betydningen av make alternativet:

`lib=lib`
Denne parameteren setter bibliotekkatalogen til `/usr/lib` heller enn `/usr/lib64` på x86_64. Det har ingen effekt på x86.

For å teste resultatene, utsted:

```
make test
```

Installer pakken:

```
make prefix=/usr lib=lib install
```

8.24.2. Innhold i Libcap

Installerte programmer: capsh, getcap, getpcaps, og setcap
Installert bibliotek: libcap.so og libpsx.so

Korte beskrivelser

capsh En skallinnpakning for å utforske og begrense kapasitetsstøtte
getcap Undersøker filfunksjoner
getpcaps Viser mulighetene for den forespurte prosessen(e)
setcap Angir filkapasiteter
`libcap` Inneholder bibliotekfunksjonene for å manipulere POSIX 1003.1e kapasiteter
`libpsx` Inneholder funksjoner for å støtte POSIX semantikk for syscalls knyttet til pthread biblioteket

8.25. Shadow-4.11.1

Shadow pakken inneholder programmer for håndtering av passord på en sikker måte.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 49 MB

8.25.1. Installasjon av Shadow

Note

Hvis du ønsker å håndheve bruken av sterke passord, se *https://www.linuxfromscratch.org/blfs/view/stable-systemd/postlfs/cracklib.html* for installasjon av CrackLib før du bygger Shadow. Legg så til *`--with-libcrack`* til **configure** kommandoen under.

Deaktiver installasjonen av **groups** programmet og man sidene, ettersom Coreutils gir en bedre versjon. Også, forhindre installasjon av man sider som allerede var installert i Section 8.3, “Man-pages-5.13”:

```
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 / /'   {} \;
```

I stedet for å bruke standard *crypt* metoden, bruk den sikrere *SHA-512* metode for passordkryptering, som også tillater passord lengre enn 8 tegn. Det er også nødvendig å endre det foreldede `/var/spool/mail` plasseringen for brukerpostbokser som Shadow bruker som standard til `/var/mail` stedet som brukes for øyeblikket. Og, kvitte seg med `/bin` og `/sbin` fra `PATH`, siden de ganske enkelt er symbolske lenker til motparten i `/usr`.

Note

Hvis `/bin` og/eller `/sbin` foretrekkes å være i `PATH` av en eller annen grunn, endre `PATH` i `.bashrc` etter at LFS er bygget.

```
sed -e 's:#ENCRYPT_METHOD DES:ENCRYPT_METHOD SHA512:' \
    -e 's:/var/spool/mail:/var/mail:'                \
    -e '/PATH=/{s@/sbin:@@;s@/bin:@@}'               \
    -i etc/login.defs
```

Note

Hvis du valgte å bygge Shadow med støtte for Cracklib, kjør følgende:

```
sed -i 's:DICTPATH.*:DICTPATH\t/lib/cracklib/pw_dict:' etc/login.defs
```

Forbered Shadow for kompilering:

```
touch /usr/bin/passwd
./configure --sysconfdir=/etc \
            --disable-static  \
            --with-group-name-max-length=32
```

Betydningen av konfigureringsalternativet:

touch /usr/bin/passwd
Filen `/usr/bin/passwd` må eksistere fordi plasseringen er hardkodet i noen programmer, og hvis den ikke eksisterer, er ikke standardplasseringen riktig.

`--with-group-name-max-length=32`
Maksimalt brukernavn er 32 tegn. Gjør det maksimale gruppenavnet det samme.

Kompiler pakken:

```
make
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make exec_prefix=/usr install
make -C man install-man
```

8.25.2. Konfigurerer Shadow

Denne pakken inneholder verktøy for å legge til, endre og slette brukere og grupper; angi og endre passordene deres; og utføre annen administrativ oppgaver. For en fullstendig forklaring av hva *passordskygge* betyr, se `doc/HOWTO` filen i den utpakkede kildetreet. Hvis du bruker Shadow-støtte, husk at programmer som trenger å bekrefte passord (skjermbehandlere, FTP-programmer, pop3-nisser, etc.) må være Shadow-kompatibel. Det vil si at de må kunne jobbe med skyggelagte passord.

For å aktivere skyggelagte passord, kjør følgende kommando:

```
pwconv
```

For å aktivere skyggelagte gruppepassord, kjør:

```
grpconv
```

Shadows standardkonfigurasjon for **useradd** verktøyet har noen forbehold som trenger litt forklaring. Først standard handling for **useradd** verktøyet er å lage brukeren og en gruppe med samme navn som brukeren. Som standard begynner bruker-ID (UID) og gruppe ID-numre (GID) med 1000. Dette betyr at hvis du ikke sender parametere til **useradd**, hver bruker vil være medlem av en unik gruppe på systemet. Hvis denne oppførselen er uønsket, trenger du å sende enten *`-g`* eller *`-N`* parameter til **useradd** eller for å endre innstillingen for *`USERGROUPS_ENAB`* i `/etc/login.defs`. Se `useradd(8)` for mer informasjon.

For det andre, for å endre standardparametrene, filen `/etc/default/useradd` må lages og skreddersys for å passe dine spesielle behov. Lag den med:

```
mkdir -p /etc/default
useradd -D --gid 999
```

`/etc/default/useradd` Parameterforklaringer

`GROUP=999`
Denne parameteren setter begynnelsen på gruppenumrene som brukes i `/etc/group` filen. Den spesielle verdien 999 kommer fra *`--gid`* parameter ovenfor. Du kan endre den til alt du ønsker. Noter at **useradd** vil aldri gjenbruke

en UID eller GID. Hvis nummeret som er identifisert i denne parameteren brukes, vil det bruke neste ledige nummer. Merk også at hvis du ikke har en gruppe med en ID lik dette nummeret på systemet ditt første gang du bruker **useradd** uten *-g* parameteren, du vil få en melding på terminalen som sier: `useradd: unknown GID 999`, selv om kontoen er riktig opprettet. Det er derfor vi har opprettet gruppen `users` med denne gruppe-IDen i Section 7.6, “Opprette essensielle filer og symbolkoblinger”.

`CREATE_MAIL_SPOOL=yes`

Denne parameteren gjør at **useradd** lager en postboksfil for den nyopprettede brukeren. **useradd** vil gjøre gruppe eierskap av denne filen til `mail` gruppe med 0660 tillatelser. Hvis du foretrekker at disse postboksfilene ikke blir laget av **useradd**, utsted følgende kommando:

```
sed -i '/MAIL/s/yes/no/' /etc/default/useradd
```

8.25.3. Sette root passordet

Velg et passord for brukeren *root* og sett det ved å kjøre:

```
passwd root
```

8.25.4. Innhold i Shadow

Installerte programmer:	chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, getsubids, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgidmap, newgrp, newuidmap, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (link to newgrp), su, useradd, userdel, usermod, vigr (link to vipw), og vipw
Installert mappe:	/etc/default
Installerte biblioteker:	libsubid.so

Korte beskrivelser

chage	Brukes til å endre maksimalt antall dager mellom obligatoriske passordendringer
chfn	Brukes til å endre en brukers fulle navn og annen informasjon
chgpasswd	Brukes til å oppdatere gruppepassord i skriptmodus
chpasswd	Brukes til å oppdatere brukerpassord i skriptmodus
chsh	Brukes til å endre en brukers standard påloggingsskall
expiry	Sjekker og håndhever gjeldende retningslinjer for passordutløp
faillog	Brukes til å undersøke loggen over påloggingsfeil, for å sette et maksimum antall feil før en konto blokkeres, eller for å tilbakestille antall feil
getsubids	Brukes til å liste de underordnede id-områdene for en bruker
gpasswd	Brukes til å legge til og slette medlemmer og administratorer til grupper
groupadd	Oppretter en gruppe med det gitte navnet
groupdel	Sletter gruppen med oppgitt navn
groupmems	Lar en bruker administrere sin egen gruppemedlemsliste uten krav om superbrukerprivilegier.
groupmod	Brukes til å endre den gitte gruppens navn eller GID
grpck	Verifiserer integriteten til gruppefilene `/etc/group` og `/etc/gshadow`

grpconv Oppretter eller oppdaterer skyggegruppefilen fra den normale gruppefilen

grpunconv Oppdaterer `/etc/group` fra `/etc/gshadow` og sletter deretter sistnevnte

lastlog Rapporterer siste pålogging for alle brukere eller av en gitt bruker

login Brukes av systemet for å la brukere logge på

logoutd Er en nisse som brukes til å håndheve restriksjoner på påloggingstid og portene

newgidmap Brukes til å angi gid-tilordning av et brukernavnområde

newgrp Brukes til å endre gjeldende GID under en påloggingsøkt

newuidmap Brukes til å angi uid-tilordning av et brukernavnområde

newusers Brukes til å opprette eller oppdatere en hel serie med brukerkontoer

nologin Viser en melding om at en konto ikke er tilgjengelig; den er designet til å brukes som standard skall for kontoer som er blitt deaktivert

passwd Brukes til å endre passordet for en bruker- eller gruppekonto

pwck Verifiserer integriteten til passordfilene `/etc/passwd` og `/etc/shadow`

pwconv Oppretter eller oppdaterer skyggepassordfilen fra den normale passordfilen

pwunconv Oppdaterer `/etc/passwd` fra `/etc/shadow` og sletter deretter sistnevnte

sg Utfører en gitt kommando mens brukerens GID er satt til den gitte gruppen

su Kjører et skall med erstatningsbruker- og gruppe-IDer

useradd Oppretter en ny bruker med det gitte navnet, eller oppdaterer standard informasjon om ny bruker

userdel Sletter den gitte brukerkontoen

usermod Brukes til å endre den gitte brukerens påloggingsnavn, bruker identifikasjon (UID), skall, startgruppe, hjemmekatalog, etc.

vigr Redigerer `/etc/group` eller `/etc/gshadow` filer

vipw Redigerer `/etc/passwd` eller `/etc/shadow` filer

`libsubid` bibliotek for å prosessere underordnede id-områder for brukere

8.26. GCC-11.2.0

GCC pakken inneholder GNU kompilatorsamlingen, som inkluderer C og C++ kompilatorene.

Omtrentlig byggetid: 153 SBU (med testene)
Nødvendig diskplass: 4.3 GB

8.26.1. Installasjon av GCC

Først må du fikse et problem som forhindrer `libasan.a` å bygge denne pakken med Glibc-2.34 eller nyere:

```
sed -e '/static.*SIGSTKSZ/d' \
    -e 's/return kAltStackSize/return SIGSTKSZ * 4/' \
    -i libsanitizer/sanitizer_common/sanitizer_posix_libcdep.cpp
```

Hvis du bygger på x86_64, endre standard katalognavn for 64-bit bibliotekene til “lib”:

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

GCC dokumentasjonen anbefaler å bygge GCC i en dedikert byggemappe:

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

Forbered GCC for kompilering:

```
../configure --prefix=/usr            \
             LD=ld                    \
             --enable-languages=c,c++ \
             --disable-multilib       \
             --disable-bootstrap      \
             --with-system-zlib
```

Merk at for andre programmeringsspråk er det noen forutsetninger som ikke er tilgjengelig ennå. Se *BLFS Bokens GCC side* for instruksjoner om hvordan du bygger alle GCCs støttede språk.

Betydningen av de nye konfigureringsparametrene:

`LD=ld`
Denne parameteren gjør at konfigureringsskriptet bruker ld installert av binutils bygget tidligere i dette kapittelet, i stedet for den kryssbygde versjonen som ellers ville blitt brukt.

`--with-system-zlib`
Denne bryteren forteller GCC å koble til den systeminstallerte kopien av zlib biblioteket, i stedet for sin egen interne kopi.

Kompiler pakken:

```
make
```

Important

I denne delen vurderes testpakken for GCC å være viktig, men det tar lang tid. Førstegangsbyggere oppfordres til ikke å hoppe over dette. Tiden for å kjøre testene kan bli redusert betydelig ved å legge til -jx i make kommandoen nedenfor hvor x er antall kjerner på systemet ditt.

Et sett med tester i GCC testpakken er kjent for å bruke opp standard stabel (stack), så øk stabelstørrelsen før du kjører testene:

```
ulimit -s 32768
```

Test resultatene som en ikke-privilegert bruker, men ikke stopp ved feil:

```
chown -Rv tester .
su tester -c "PATH=$PATH make -k check"
```

For å motta et sammendrag av resultatene til testpakken, kjør:

```
../contrib/test_summary
```

For bare sammendragene, kanaliser utdataene gjennom **`grep -A7 Summ`**.

Resultatene kan sammenlignes med de som ligger på *https://www.linuxfromscratch.org/lfs/build-logs/11.1/* og *https://gcc.gnu.org/ml/gcc-testresults/*.

Åtte tester relatert til analyzer er kjent for å mislykkes.

En test med navnet `asan_test.C` er kjent for mislykkes.

I libstdc++ en test med navnet `49745.cc` er kjent for å mislykkes fordi deklarasjonsavhengighetene i glibc er endret.

I libstdc++ er en numpunct test og seks tester relatert til get_time kjent for å mislykkes. Dette er fordi lokalitetsdefinisjonene i glibc er endret, men libstdc++ støtter for øyeblikket ikke disse endringene.

Noen få uventede feil kan ikke alltid unngås. GCC utviklerne er vanligvis klar over disse problemene, men har ikke løst dem ennå. Med mindre testresultatene er svært forskjellige fra de på URLen ovenfor, er det trygt å fortsette.

Installer pakken og fjern en unødvendig mappe:

```
make install
rm -rf /usr/lib/gcc/$(gcc -dumpmachine)/11.2.0/include-fixed/bits/
```

GCC byggekatalogen eies av `tester` nå og eierskapet til den installerte deklarasjonsmappen (og dens innhold) vil være feil. Endre eierskapet til `root` bruker og gruppe:

```
chown -v -R root:root \
    /usr/lib/gcc/*linux-gnu/11.2.0/include{,-fixed}
```

Lag en symbolkobling som kreves av *FHS* av "historiske" grunner.

```
ln -svr /usr/bin/cpp /usr/lib
```

Legg til en kompatibilitetssymbolkobling for å aktivere byggeprogrammer med optimalisering av koblingstid (LTO):

```
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/11.2.0/liblto_plugin.so \
        /usr/lib/bfd-plugins/
```

Nå som vår endelige verktøykjede er på plass, er det viktig å sikre at kompilering og kobling vil fungere som forventet. Dette gjør vi ved å utføre noen sunnhetssjekker:

```
echo 'int main(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

Det skal ikke være noen feil, og utdataen av den siste kommandoen vil være (som gir rom for plattformspesifikke forskjeller i det dynamiske linkernavnet):

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

Sørg nå for at vi er konfigurert til å bruke de riktige startfilene:

```
grep -o '/usr/lib.*/crt[1in].*succeeded' dummy.log
```

Utdata fra den siste kommandoen skal være:

```
/usr/lib/gcc/x86_64-pc-linux-gnu/11.2.0/../../../../lib/crt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/11.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/11.2.0/../../../../lib/crtn.o succeeded
```

Avhengig av maskinarkitekturen din, kan ovenstående avvike litt. Forskjellen vil være navnet på mappen etter `/usr/lib/gcc`. Det viktige å se etter her er det at **gcc** har funnet alle tre `crt*.o` filene under `/usr/lib` mappen.

Kontroller at kompilatoren søker etter riktige deklarasjons filer:

```
grep -B4 '^ /usr/include' dummy.log
```

Denne kommandoen skal returnere følgende utdata:

```
#include <...> search starts here:
 /usr/lib/gcc/x86_64-pc-linux-gnu/11.2.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/11.2.0/include-fixed
 /usr/include
```

Igjen, mappen oppkalt etter måltripletten kan være annerledes enn de ovennevnte, avhengig av systemarkitekturen.

Deretter kontrollerer du at den nye linkeren brukes med de riktige søkebanene:

```
grep 'SEARCH.*/usr/lib' dummy.log |sed 's|; |\n|g'
```

Referanser til stier som har komponenter med '-linux-gnu' bør ignoreres, men ellers skal utdataene fra den siste kommandoen være:

```
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");
```

Et 32-bits system kan se noen forskjellige kataloger. For eksempel her er utdata fra en i686-maskin:

```
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");
```

Deretter må du kontrollere at vi bruker riktig libc:

```
grep "/lib.*/libc.so.6 " dummy.log
```

Utdata fra den siste kommandoen skal være:

```
attempt to open /usr/lib/libc.so.6 succeeded
```

Sørg for at GCC bruker riktig dynamisk linker:

```
grep found dummy.log
```

Utdataene fra den siste kommandoen skal være (tillater plattformspesifikke forskjeller i dynamisk linkernavn):

```
found ld-linux-x86-64.so.2 at /usr/lib/ld-linux-x86-64.so.2
```

Hvis utdataene ikke vises som vist ovenfor eller ikke mottas i det hele tatt, så er det noe alvorlig galt. Undersøk og gå gjennom trinn for trinn, for å finne ut hvor problemet er og rette det. Alle problemer må løses før du fortsetter med prosessen.

Når alt fungerer som det skal, rydd opp i testfilene:

```
rm -v dummy.c a.out dummy.log
```

Til slutt flytter du en feilplassert fil:

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

8.26.2. Innhold i GCC

Installerte programmer: c++, cc (lenker til gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, gcov, gcov-dump, gcov-tool, og lto-dump

Installerte biblioteker: libasan.{a,so}, libatomic.{a,so}, libcc1.so, libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, libitm.{a,so}, liblsan.{a,so}, liblto_plugin.so, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so}, libstdc++fs.a, libsupc++.a, libtsan.{a,so}, og libubsan.{a,so}

Installerte mapper: /usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc, og /usr/share/gcc-11.2.0

Korte beskrivelser

c++ C++ kompilatoren

cc	C kompilatoren
cpp	C-forprosessoren; den brukes av kompilatoren for å utvide #include, #define og lignende utsagn i kildefilene
g++	C++ kompilatoren
gcc	C kompilatoren
gcc-ar	En innpakning rundt **ar** som legger til et programtillegg til kommandolinjen. Dette programmet brukes kun å legge til "koblingstidsoptimalisering (LTO)" og er ikke nyttig med standard byggealternativer
gcc-nm	En innpakning rundt **nm** som legger til et programtillegg til kommandolinjen. Dette programmet brukes kun å legge til "koblingstidsoptimalisering (LTO" og er ikke nyttig med standard byggealternativer
gcc-ranlib	En innpakning rundt **ranlib** som legger til et programtillegg til kommandolinjen. Dette programmet brukes kun å legge til "koblingstidsoptimalisering (LTO" og er ikke nyttig med standard byggealternativer
gcov	Et dekningstestverktøy; den brukes til å analysere programmer å bestemme hvor optimaliseringer vil ha størst effekt
gcov-dump	Frakoblet gcda og gcno profildumpverktøy
gcov-tool	Frakoblet gcda profilbehandlingsverktøy
lto-dump	Verktøy for dumping av objektfiler produsert av GCC med LTO aktivert
`libasan`	Kjøretidsbiblioteket for adresserensing
`libatomic`	GCC atomic innebygde kjøretidsbibliotek
`libcc1`	C forbehandlingsbiblioteket
`libgcc`	Inneholder kjøretidsstøtte for **gcc**
`libgcov`	Dette biblioteket er koblet inn i et program når GCC blir instruert om å aktivere profilering
`libgomp`	GNU implementering av OpenMP API for multiplattform parallellprogrammering med delt minne i C/C++ og Fortran
`libitm`	GNU transaksjonsminnebiblioteket
`liblsan`	Leak Sanitizer kjøretidsbibliotek
`liblto_plugin`	GCC sitt LTO programtillegg som lar binutils behandle objektfiler produsert av GCC med LTO aktivert
`libquadmath`	GCC Quad Precision Math Library API
`libssp`	Inneholder rutiner som støtter GCCs stabelknusende beskyttelses funksjonalitet
`libstdc++`	Standard C++ biblioteket
`libstdc++fs`	ISO/IEC TS 18822:2015 filsystembiblioteky
`libsupc++`	Gir støttende rutiner for C++ programmeringsspråk
`libtsan`	Thread Sanitizer kjøretidsbibliotek
`libubsan`	Undefined Behavior Sanitizer kjøretidsbibliotek

8.27. Pkg-config-0.29.2

Pakken pkg-config inneholder et verktøy for å sende inkluderingsbanen og/eller bibliotekstier for å bygge verktøy under konfigurerings- og makefasene av pakkeinstallasjoner.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 29 MB

8.27.1. Installasjon av Pkg-config

Forbered Pkg-config for kompilering:

```
./configure --prefix=/usr              \
            --with-internal-glib       \
            --disable-host-tool        \
            --docdir=/usr/share/doc/pkg-config-0.29.2
```

Betydningen av de nye konfigureringsalternativene:

`--with-internal-glib`
Dette vil tillate pkg-config å bruke sin interne versjon av Glib fordi en ekstern versjon ikke er tilgjengelig i LFS.

`--disable-host-tool`
Dette alternativet deaktiverer opprettelsen av en uønsket hard lenke til pkg-config programmet.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.27.2. Innhold i Pkg-config

Installert program: pkg-config
Installert mappe: /usr/share/doc/pkg-config-0.29.2

Korte beskrivelser

pkg-config Returnerer metainformasjon for det angitte biblioteket eller pakken

8.28. Ncurses-6.3

Ncurses pakken inneholder biblioteker for terminaluavhengig håndtering av karakterskjermer.

Omtrentlig byggetid: 0.4 SBU
Nødvendig diskplass: 45 MB

8.28.1. Installasjon av Ncurses

Forbered Ncurses for kompilering:

```
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            --with-shared           \
            --without-debug         \
            --without-normal        \
            --enable-pc-files       \
            --enable-widec          \
            --with-pkg-config-libdir=/usr/lib/pkgconfig
```

Betydningen av de nye konfigureringsalternativene:

`--enable-widec`

Denne bryteren gjør at biblioteker med store tegn (f.eks., `libncursesw.so.6.3`) bygges i stedet for vanlige (f.eks., `libncurses.so.6.3`). Disse brede tegnbibliotekene er brukbare i både multibyte og tradisjonelle 8-biters lokaliteter, mens vanlige biblioteker fungerer som de skal bare i 8-biters lokaliteter. Brede karakterer og normale biblioteker er kildekompatibel, men ikke binærkompatibel.

`--enable-pc-files`

Denne bryteren genererer og installerer .pc-filer for pkg-config.

`--without-normal`

Denne bryteren deaktiverer bygging og installasjon av de fleste statiske biblioteker.

Kompiler pakken:

```
make
```

Denne pakken har en testpakke, men den kan bare kjøres etter at pakken er installert. Testene ligger i `test/` mappen. Se `README` filen i den mappen for ytterligere detaljer.

Installasjonen av denne pakken vil overskrive `libncursesw.so.6.3` . Det kan krasje skallprosessen som bruker kode og data fra bibliotekfilen. Installer pakken med `DESTDIR`, og bytt ut bibliotekfilen riktig med **install** kommandoen. Et ubrukelig statisk arkiv som ikke håndteres av **configure** er også fjernet:

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/libncursesw.so.6.3 /usr/lib
rm -v  dest/usr/lib/{libncursesw.so.6.3,libncurses++w.a}
cp -av dest/* /
```

Mange applikasjoner forventer fortsatt at lenkeren skal kunne finne Ncurses biblioteker med non-wide karakterer. Lur slike applikasjoner til å koble til biblioteker med brede tegn ved hjelp av symbolkoblinger og lenkerskript:

```
for lib in ncurses form panel menu ; do
    rm -vf                    /usr/lib/lib${lib}.so
    echo "INPUT(-l${lib}w)" > /usr/lib/lib${lib}.so
    ln -sfv ${lib}w.pc        /usr/lib/pkgconfig/${lib}.pc
done
```

Til slutt, sørg for at gamle programmer som ser etter `-lcurses` ved byggetiden fortsatt er byggbare:

```
rm -vf                     /usr/lib/libcursesw.so
echo "INPUT(-lncursesw)" > /usr/lib/libcursesw.so
ln -sfv libncurses.so      /usr/lib/libcurses.so
```

Hvis ønskelig, installer Ncurses dokumentasjonen:

```
mkdir -pv      /usr/share/doc/ncurses-6.3
cp -v -R doc/* /usr/share/doc/ncurses-6.3
```

Note

Instruksjonene ovenfor oppretter ikke Ncurses med non-wide tegn biblioteker siden ingen pakke installert ved kompilering fra kilder ville kobles mot dem under kjøring. Imidlertid den eneste kjente bare binær applikasjonen som kobler mot Ncurses-biblioteker med non-wide karakterer krever versjon 5. Hvis du må ha slike biblioteker på grunn av noen bare binær applikasjon eller for å være kompatibel med LSB, bygg pakken på nytt med følgende kommandoer:

```
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
```

8.28.2. Innhold i Ncurses

Installerte programmer: captoinfo (lenker til tic), clear, infocmp, infotocap (lenker til tic), ncursesw6-config, reset (lenker til tset), tabs, tic, toe, tput, og tset

Installerte biblioteker: libcursesw.so (symbollink og skript som lenker til libncursesw.so), libformw.so, libmenuw.so, libncursesw.so, libpanelw.so, og deres non-wide karakters motstykker uten "w" i biblioteknavnene.

Installerte mapper: /usr/share/tabset, /usr/share/terminfo, og /usr/share/doc/ncurses-6.3

Korte beskrivelser

captoinfo Konverterer en termcap beskrivelse til en terminfo beskrivelse

clear	Tømmer skjermen hvis mulig
infocmp	Sammenligner eller skriver ut terminfo beskrivelser
infotocap	Konverterer en terminfo beskrivelse til en termcap beskrivelse
ncursesw6-config	Gir konfigurasjonsinformasjon for ncurses
reset	Reinitialiserer en terminal til standardverdiene
tabs	Fjerner og setter tabulatorstopp på en terminal
tic	Terminfo entry-description-kompilatoren som oversetter en terminfo fil fra kildeformat til det binære formatet som trengs for ncurses biblioteksrutiner [En terminfo fil inneholder informasjon om egenskapene til en bestemt terminal.]
toe	Viser alle tilgjengelige terminaltyper, med primærnavn og beskrivelse for hver
tput	Gjør verdiene til terminalavhengige funksjoner tilgjengelig for skallet; den kan også brukes til å tilbakestille eller initialisere en terminal eller rapporter det lange navnet
tset	Kan brukes til å initialisere terminaler
`libcursesw`	En lenke til `libncursesw`
`libncursesw`	Inneholder funksjoner for å vise tekst på mange komplekse måter på en terminalskjerm; et godt eksempel på bruken av disse funksjonene er menyen som vises under kjernens **make menuconfig**
`libformw`	Inneholder funksjoner for å implementere skjemaer
`libmenuw`	Inneholder funksjoner for å implementere menyer
`libpanelw`	Inneholder funksjoner for å implementere paneler

8.29. Sed-4.8

Sed pakken inneholder en dataflyt (stream) redigerer.

Omtrentlig byggetid: 0.4 SBU
Nødvendig diskplass: 31 MB

8.29.1. Installation of Sed

Forbered Sed for kompilering:

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

Kompiler pakken og generer HTML dokumentasjonen:

```
make
make html
```

For å teste resultatene, utsted:

```
chown -Rv tester .
su tester -c "PATH=$PATH make check"
```

Installer pakken og dens dokumentasjon:

```
make install
install -d -m755           /usr/share/doc/sed-4.8
install -m644 doc/sed.html /usr/share/doc/sed-4.8
```

8.29.2. Innhold i Sed

Installert program: sed
Installert mappe: /usr/share/doc/sed-4.8

Short Descriptions

sed Filtrerer og transformerer tekstfiler i en enkelt omgang

8.30. Psmisc-23.4

Psmisc pakken inneholder programmer for å vise informasjon om kjørende prosesser.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 5.6 MB

8.30.1. Installasjon av Psmisc

Forbered Psmisc for kompilering:

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

Kompiler pakken:

```
make
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make install
```

8.30.2. Innhold i Psmisc

Installerte programmer: fuser, killall, peekfd, prtstat, pslog, pstree, og pstree.x11 (lenker til pstree)

Korte beskrivelser

fuser Rapporterer prosessIDene (PIDene) til prosesser som bruker de gitte filer eller filsystemer

killall Dreper prosesser ved navn; den sender et signal til alle prosesser som kjører noen av de gitte kommandoene

peekfd Se på filbeskrivelser for en prosess som kjører, gitt dens PID

prtstat Skriver ut informasjon om en prosess

pslog Rapporterer gjeldende loggbane for en prosess

pstree Viser kjørende prosesser som et tre

pstree.x11 Samme som **pstree**, bortsett fra at den venter på bekreftelse før den avslutter

8.31. Gettext-0.21

Gettext pakken inneholder verktøy for internasjonalisering og lokalisering. Disse gjør at programmer kan kompileres med NLS (Lokal Språk Støtte), slik at de kan sende ut meldinger i brukerens lokale språkformat.

Omtrentlig byggetid: 2.7 SBU
Nødvendig diskplass: 233 MB

8.31.1. Installasjon av Gettext

Forbered Gettext for kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.21
```

Kompiler pakken:

```
make
```

For å teste resultatene (dette tar lang tid, rundt 3 SBU), utsted:

```
make check
```

Installer pakken:

```
make install
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

8.31.2. Innhold i Gettext

Installerte programmer: autopoint, envsubst, gettext, gettext.sh, gettextize, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin, og xgettext
Installerte biblioteker: libasprintf.so, libgettextlib.so, libgettextpo.so, libgettextsrc.so, libtextstyle.so, og preloadable_libintl.so
Installerte mapper: /usr/lib/gettext, /usr/share/doc/gettext-0.21, /usr/share/gettext, og /usr/share/gettext-0.19.8

Korte beskrivelser

autopoint Kopierer standard Gettext infrastrukturfiler til en kildepakke

envsubst Erstatter miljøvariabler i skallformatstrenger

gettext Oversetter en melding på det opprinnelige språket til brukerens språk ved å slå opp oversettelsen i en meldingskatalog

gettext.sh Fungerer først og fremst som et skallfunksjonsbibliotek for gettext

gettextize Kopierer alle standard Gettext filer til den gitte mappens toppnivå til en pakke for å begynne å internasjonalisere den

msgattrib Filtrerer meldingene i en oversettelsesmappe i henhold til deres attributter og manipulerer attributtene

msgcat	Sammenslår og slår sammen de gitte `.po` filene
msgcmp	Sammenligner to `.po` filer for å sjekke at begge inneholder samme sett med msgid strenger
msgcomm	Finner meldingene som er felles for de gitte `.po` filene
msgconv	Konverterer en oversettelseskatalog til en annet tegnkoding
msgen	Oppretter en engelsk oversettelseskatalog
msgexec	Bruker en kommando på alle oversettelser av en oversettelsesmappe
msgfilter	Bruker et filter på alle oversettelser av en oversettelsesmappe
msgfmt	Genererer en binær meldingskatalog fra en oversettelsesmappe
msggrep	Trekker ut alle meldinger fra en oversettelseskatalog som samsvarer med et gitt mønster eller tilhører noen gitte kildefiler
msginit	Oppretter en ny `.po` fil, initialisere metainformasjonen med verdier fra brukerens miljø
msgmerge	Kombinerer to rå oversettelser til én enkelt fil
msgunfmt	Dekompilerer en binær meldingskatalog til rå oversettelsestekst
msguniq	Forener dupliserte oversettelser i en oversettelsesmappe
ngettext	Viser oversettelser på morsmål av en tekstmelding hvis grammatisk form avhenger av et tall
recode-sr-latin	Omkoder serbisk tekst fra kyrillisk til latinsk skrift
xgettext	Trekker ut de oversettbare meldingslinjene fra den gitte kildefilen for å lage den første oversettelsesmalen
`libasprintf`	definerer *autosprintf* klassen, som gir C formaterte utdatarutiner som kan brukes i C++ programmer, for bruk med <*string*> strenger og <*iostream*> dataflyt
`libgettextlib`	et privat bibliotek som inneholder vanlige rutiner som brukes av ulike Gettext programmer; disse er ikke beregnet for generelt bruk
`libgettextpo`	Brukes til å skrive spesialiserte programmer som behandler `.po` filer; dette biblioteket brukes når standardapplikasjonene som ble levert med Gettext (som f.eks **msgcomm**, **msgcmp**, **msgattrib**, og **msgen**) ikke er tilstrekkelig
`libgettextsrc`	et privat bibliotek som inneholder vanlige rutiner som brukes av ulike Gettext programmer; disse er ikke beregnet for generelt bruk
`libtextstyle`	Tekststilbibliotek
`preloadable_libintl`	Et bibliotek, ment å brukes av LD_PRELOAD som assisterer `libintl` i å logge uoversatte meldinger

8.32. Bison-3.8.2

Bison pakken inneholder en parsergenerator.

Omtrentlig byggetid: 6.3 SBU
Nødvendig diskplass: 53 MB

8.32.1. Installasjon av Bison

Forbered Bison for kompilering:

```
./configure --prefix=/usr --docdir=/usr/share/doc/bison-3.8.2
```

Kompiler pakken:

```
make
```

For å teste resultatene (about 5.5 SBU), utsted:

```
make check
```

Installer pakken:

```
make install
```

8.32.2. Innholdet i Bison

Installerte programmer: bison og yacc
Installerte biblioteker: liby.a
Installert katalog: /usr/share/bison

Korte beskrivelser

bison Genererer, fra en rekke regler, et program for å analysere struktur av tekstfiler; Bison er en erstatning for Yacc (Yet Another Compiler Compiler)

yacc En innpakning for **bison**,ment for programmer som fortsatt kaller **yacc** i stedet for **bison**; den kaller **bison** med *-y* alternativet

`liby` Yacc biblioteket som inneholder implementeringer av Yacc kompatibel `yyerror` og `main` funksjoner; dette biblioteket er normalt lite nyttig, men POSIX krever det

8.33. Grep-3.7

Grep pakken inneholder programmer for å søke gjennom innholdet i filer.

Omtrentlig byggetid: 0.9 SBU
Nødvendig diskplass: 36 MB

8.33.1. Installasjon av Grep

Forbered Grep for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.33.2. Innhold i Grep

Installerte programmer: egrep, fgrep, and grep

Korte beskrivelser

egrep Skriver ut linjer som samsvarer med et utvidet regulært uttrykk

fgrep Skriver ut linjer som samsvarer med en liste over faste strenger

grep Skriver ut linjer som samsvarer med et grunnleggende regulært uttrykk

8.34. Bash-5.1.16

Bash pakken inneholder Bourne-Again Skallet (Bourne-Again SHell).

Omtrentlig byggetid: 1.5 SBU
Nødvendig diskplass: 50 MB

8.34.1. Installasjon av Bash

Forbered Bash for kompilering:

```
./configure --prefix=/usr                      \
            --docdir=/usr/share/doc/bash-5.1.16 \
            --without-bash-malloc              \
            --with-installed-readline
```

Betydningen av det nye konfigureringsalternativet:

`--with-installed-readline`
Dette alternativet forteller Bash å bruke `readline` biblioteket som allerede er installert på systemet i stedet for å bruke sin egen readline versjon.

Kompiler pakken:

```
make
```

Hopp ned til "Installer pakken" hvis du ikke kjører testpakken.

For å forberede testene, sørg for at brukeren `tester` kan skrive til kildetreet:

```
chown -Rv tester .
```

Testpakken til pakken er designet for å kjøres som en ikke-root bruker som eier terminalen koblet til standardinngang. For å tilfredsstille krav, skap en ny pseudoterminal ved hjelp av Expect og kjør testene som bruker `tester`:

```
su -s /usr/bin/expect tester << EOF
set timeout -1
spawn make tests
expect eof
lassign [wait] _ _ _ value
exit $value
EOF
```

Installer pakken:

```
make install
```

Kjør den nylig kompilerte **bash** programmet (erstatter det som kjøres for øyeblikket):

```
exec /usr/bin/bash --login
```

8.34.2. Innholdet i Bash

Installerte programmer: bash, bashbug, og sh (linker til bash)
Installert mappe: /usr/include/bash, /usr/lib/bash, og /usr/share/doc/bash-5.1.16

Korte beskrivelser

bash En mye brukt kommandotolk; den utfører mange typer av utvidelser og erstatninger på en gitt kommandolinje før kjøringen gjøres , og dette gjør dermed denne tolken til et kraftig verktøy

bashbug Et skallskript for å hjelpe brukeren med å skrive og sende standard formaterte feilrapporter vedrørende **bash**

sh En symbolsk lenke til **bash** programmet; når det påkalles som **sh**, **bash** prøver å etterligne oppstartadferd av historiske versjoner av **sh** så nært som mulig, samtidig som den samsvarer med POSIX standarden også

8.35. Libtool-2.4.6

Libtool pakken inneholder GNU generiske bibliotekstøtteskript. Det omslutter kompleksiteten ved å bruke delte biblioteker i en konsistent, overførbart grensesnitt.

Omtrentlig byggetid: 1.5 SBU
Nødvendig diskplass: 43 MB

8.35.1. Installasjon av Libtool

Forbered Libtool for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Note

Testtiden for libtool kan reduseres betydelig på et system med flere kjerner. For å gjøre dette, legg til **TESTSUITEFLAGS=-j<N>** til linjen over. For eksempel kan bruk av -j4 redusere testtiden med over 60 prosent.

Fem tester er kjent for å mislykkes i LFS byggemiljøet pga en sirkulær avhengighet, men alle tester består hvis de sjekkes på nytt etter at automake er installert.

Installer pakken:

```
make install
```

Fjern et ubrukelig statisk bibliotek:

```
rm -fv /usr/lib/libltdl.a
```

8.35.2. Innhold i Libtool

Installerte programmer: libtool og libtoolize
Installerte biblioteker: libltdl.so
Installerte mapper: /usr/include/libltdl og /usr/share/libtool

Korte beskrivelser

libtool Tilbyr generaliserte støttetjenester for bibliotekbygging

libtoolize Gir en standard måte å legge til **libtool** støtte til en pakke

`libltdl` Skjuler de forskjellige vanskelighetene med dlopening biblioteker

8.36. GDBM-1.23

GDBM pakken inneholder GNU Database Manager. Det er et bibliotek av databasefunksjoner som bruker utvidbar hashing og fungerer lignende som standard UNIX dbm. Biblioteket gir primitiver for lagring av nøkkel/data par, søker og henter dataene etter nøkkelen og sletter en nøkkel sammen med sine data.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 13 MB

8.36.1. Installasjon av GDBM

Forbered GDBM for kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --enable-libgdbm-compat
```

Betydningen av konfigureringsalternativet:

`--enable-libgdbm-compat`
Denne bryteren gjør det mulig å bygge libgdbm kompatibilitetsbiblioteket. Noen pakker utenfor LFS kan kreve det eldre DBM rutiner gir.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.36.2. Innhold i GDBM

Installerte programmer: gdbm_dump, gdbm_load, og gdbmtool
Installerte biblioteker: libgdbm.so og libgdbm_compat.so

Korte beskrivelser

gdbm_dump Dumper en GDBM database til en fil

gdbm_load Gjenoppretter en GDBM database fra en dumpfil

gdbmtool Tester og modifiserer en GDBM database

`libgdbm` Inneholder funksjoner for å manipulere en hashet database

`libgdbm_compat` Kompatibilitetsbibliotek som inneholder eldre DBM funksjoner

8.37. Gperf-3.1

Gperf genererer en perfekt hashfunksjon fra et nøkkelsett.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 6.0 MB

8.37.1. Installasjon av Gperf

Forbered Gperf for kompilering:

```
./configure --prefix=/usr --docdir=/usr/share/doc/gperf-3.1
```

Kompiler pakken:

```
make
```

Testene er kjent for å mislykkes hvis de kjører flere samtidige tester (-j alternativ større enn 1). Å teste resultatene, utsted:

```
make -j1 check
```

Installer pakken:

```
make install
```

8.37.2. Innhold i Gperf

Installert program: gperf
Installert mappe: /usr/share/doc/gperf-3.1

Korte beskrivelser

gperf Genererer en perfekt hash fra et nøkkelsett

8.38. Expat-2.4.6

Expat pakken inneholder et dataflytorientert C bibliotek for å analysere XML.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 12 MB

8.38.1. Installasjon av Expat

Forbered Expat for kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.4.6
```

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

Hvis ønskelig, installer dokumentasjonen:

```
install -v -m644 doc/*.{html,css} /usr/share/doc/expat-2.4.6
```

8.38.2. Innhold i Expat

Installert program: xmlwf
Installerte biblioteker: libexpat.so
Installert mappe: /usr/share/doc/expat-2.4.6

Korte beskrivelser

xmlwf Er et ikke-validerende verktøy for å sjekke om XML dokumenter er godt utformet

`libexpat` Inneholder API funksjoner for å analysere XML

8.39. Inetutils-2.2

Inetutils pakken inneholder programmer for grunnleggende nettverksbygging.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 30 MB

8.39.1. Installasjon av Inetutils

Forbered Inetutils for kompilering:

```
./configure --prefix=/usr        \
            --bindir=/usr/bin    \
            --localstatedir=/var \
            --disable-logger     \
            --disable-whois      \
            --disable-rcp        \
            --disable-rexec      \
            --disable-rlogin     \
            --disable-rsh        \
            --disable-servers
```

Betydningen av konfigureringsalternativene:

`--disable-logger`
Dette alternativet forhindrer Inetutils fra å installere **logger** programmet, som brukes av skript til sende meldinger til Systemlogg nissen (System Log Daemon). Ikke installer det fordi Util-linux installerer en nyere versjon.

`--disable-whois`
Dette alternativet deaktiverer byggingen av Inetutils sin **whois** klient, som er utdatert. Instruksjoner for en bedre **whois** klienten er i BLFS boken.

`--disable-r*`
Disse parameterne deaktiverer bygging av foreldede programmer som ikke burde brukes på grunn av sikkerhetsproblemer. Funksjonene som tilbys av disse programmer kan leveres av openssh pakken i BLFS boken.

`--disable-servers`
Dette deaktiverer installasjonen av de forskjellige nettverksserverne inkludert som en del av Inetutils pakken. Disse serverne anses ikke som hensiktsmessig i et grunnleggende LFS system. Noen er usikre av natur og er ansett som trygt kun på pålitelige nettverk. Merk at bedre erstatninger er tilgjengelige for mange av disse serverne.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

Flytt et program til riktig plassering:

```
mv -v /usr/{,s}bin/ifconfig
```

8.39.2. Innhold i Inetutils

Installerte programmer: dnsdomainname, ftp, ifconfig, hostname, ping, ping6, talk, telnet, tftp, og traceroute

Korte beskrivelser

dnsdomainname	Vis systemets DNS domenenavn
ftp	Er protokollprogrammet for filoverføringer
hostname	Rapporterer eller angir navnet på verten
ifconfig	Administrerer nettverksgrensesnitter
ping	Sender ekkoforespørselspakker og rapporterer hvor lenge svarene tar
ping6	En versjon av **ping** for IPv6 nettverk
talk	Brukes til å snakke med en annen bruker
telnet	Et grensesnitt til TELNET protokollen
tftp	Et trivielt filoverføringsprogram
traceroute	Sporer ruten pakkene dine tar fra verten du jobber på videre til en annen vert på et nettverk, og viser alle mellomliggende hopp (porter) underveis

8.40. Less-590

Less pakken inneholder et tekstfilviser.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 4.2 MB

8.40.1. Installasjon av Less

Forbered Less for kompilering:

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

Betydningen av konfigureringsalternativene:

`--sysconfdir=/etc`
Dette alternativet forteller at programmene som er opprettet av pakken, skal se i `/etc` for konfigurasjonen filer.

Kompiler pakken:

```
make
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make install
```

8.40.2. Innhold av Less

Installerte programmer: less, lessecho, og lesskey

Korte beskrivelser

less En filviser eller søker; den viser innholdet i det gitte fil, lar brukeren rulle, finne strenger og hoppe til merker

lessecho Trengs for å utvide metakarakterer, som f.eks * og ?, i filnavn på Unix systemer

lesskey Brukes til å spesifisere tastaturbindingene for **less**

8.41. Perl-5.34.0

Perl pakken inneholder den praktiske utvinnings og rapporterings språket (Practical Extraction and Report Language).

Omtrentlig byggetid: 9.3 SBU
Nødvendig diskplass: 226 MB

8.41.1. Installasjon av Perl

Først bruker du en oppdatering som fikser et problem fremhevet av nylige versjoner av gdbm:

```
patch -Np1 -i ../perl-5.34.0-upstream_fixes-1.patch
```

Denne versjonen av Perl bygger nå Compress::Raw::Zlib og Compress::Raw::BZip2 moduler. Som standard vil Perl bruke en intern kopi av kildene for å bygge. Utfør følgende kommando slik at Perl vil bruke bibliotekene installert på systemet:

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

For å ha full kontroll over måten Perl er satt opp på, kan du fjerne “-des” alternativer fra følgende kommando og håndplukke måten denne pakken er bygget. Alternativt kan du bruke kommandoen nøyaktig som nedenfor for å bruke standardinnstillingene som Perl automatisk oppdager:

```
sh Configure -des                                         \
             -Dprefix=/usr                                \
             -Dvendorprefix=/usr                          \
             -Dprivlib=/usr/lib/perl5/5.34/core_perl      \
             -Darchlib=/usr/lib/perl5/5.34/core_perl      \
             -Dsitelib=/usr/lib/perl5/5.34/site_perl      \
             -Dsitearch=/usr/lib/perl5/5.34/site_perl     \
             -Dvendorlib=/usr/lib/perl5/5.34/vendor_perl  \
             -Dvendorarch=/usr/lib/perl5/5.34/vendor_perl \
             -Dman1dir=/usr/share/man/man1                \
             -Dman3dir=/usr/share/man/man3                \
             -Dpager="/usr/bin/less -isR"                 \
             -Duseshrplib                                 \
             -Dusethreads
```

Betydningen av konfigureringsalternativene:

`-Dvendorprefix=/usr`
Dette sikrer at **perl** vet hvordan å fortelle pakker hvor de skal installere perl modulene sine.

`-Dpager="/usr/bin/less -isR"`
Dette sikrer at **`less`** brukes i stedet for **`more`**.

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`
Siden Groff ikke er installert ennå, **Configure** tror vi ikke ønsker man sider for Perl. Å utstede disse parametere overstyrer denne avgjørelsen.

`-Duseshrplib`
Bygger en delt libperl som trengs av noen perl moduler.

`-Dusethreads`
Bygg perl med støtte for tråder.

`-Dprivlib,-Darchlib,-Dsitelib,...`
Disse innstillingene definerer hvor Perl leter etter installerte moduler. LFS redaktørene valgte å legge dem i en katalogstruktur basert på Major.Minor-versjonen av Perl (5.34) hvilket tillater oppgradering av Perl til nyere Patch nivåer (5.34.0) uten behovet for å installere alle modulene på nytt.

Kompiler pakken:

```
make
```

For å teste resultatene (ca. 11 SBU), utsted:

```
make test
```

Installer pakken og rydd opp:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

8.41.2. Innhold i Perl

Installerte programmer:	corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.34.0 (hard link to perl), perlbug, perldoc, perlivp, perlthanks (hard link to perlbug), piconv, pl2pm, pod2html, pod2man, pod2text, pod2usage, podchecker, podselect, prove, ptar, ptardiff, ptargrep, shasum, splain, xsubpp, og zipdetails
Installerte biblioteker:	Mange som ikke alle kan listes opp her
Installert mappe:	/usr/lib/perl5

Korte beskrivelser

corelist	En kommandolinjegrensesnitt til Module::CoreList
cpan	Samhandler med Comprehensive Perl Archive Network (CPAN) fra kommandolinjen
enc2xs	Bygger en Perl utvidelse for Encode modulen fra begge Unicode karaktertilordninger eller Tcl kodingsfiler
encguess	Gjetter kodingstypen til en eller flere filer
h2ph	Konverterer `.h` C deklarasjons filer til `.ph` Perl deklarasjons filer
h2xs	Konverterer `.h` C deklarasjons filer til Perl utvidelse
instmodsh	Skallskript for å undersøke installerte Perl moduler, og kan lage en tarball fra en installert modul
json_pp	Konverterer data mellom visse inndata og utdata formater
libnetcfg	Kan brukes til å konfigurere `libnet` Perl modul
perl	Kombinerer noen av de beste egenskapene til C, **sed**, **awk** og **sh** til et singelt swiss-army språk
perl5.34.0	En hard lenke til **perl**
perlbug	Brukes til å generere feilrapporter om Perl, eller modulene som kommer med den, og sender dem
perldoc	Viser et stykke dokumentasjons i pod format som er innebygd i Perl installasjonstreet eller i et Perl skript

perlivp	Perl verifiseringsprosedyre for installasjonen; det kan brukes til bekrefte at Perl og dets biblioteker er installert riktig
perlthanks	Brukes til å generere takkemeldinger på E-post til Perl utviklere
piconv	En Perl versjon av tegnkodingskonverteren **iconv**
pl2pm	Et grovt verktøy for å konvertere Perl4 `.pl` filer til Perl5 `.pm` moduler
pod2html	Konverterer filer fra pod format til HTML format
pod2man	Konverterer pod data til formatert *roff inndata
pod2text	Konverterer pod data til formatert ASCII tekst
pod2usage	Skriver ut bruksmeldinger fra innebygde pod dokumenter i filer
podchecker	Kontrollerer syntaksen til dokumentasjonsfiler i podformat
podselect	Viser valgte deler av poddokumentasjonen
prove	Kommandolinjeverktøy for å kjøre tester mot Test::Harness moduler
ptar	Et **tar** likt program skrevet i Perl
ptardiff	Et Perl program som sammenligner et ekstrahert arkiv med et uekstrahert
ptargrep	Et Perl program som bruker mønstertilpasning på innholdet av filer i et tararkiv
shasum	Skriver ut eller kontrollerer SHA sjekksummer
splain	Brukes til å fremtvinge detaljert advarselsdiagnostikk i Perl
xsubpp	Konverterer Perl XS-kode til C-kode
zipdetails	Viser detaljer om den interne strukturen til en Zip-fil

8.42. XML::Parser-2.46

XML::Parser modulen er et Perl grensesnitt til James Clarks XML-parser, Expat.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 2.4 MB

8.42.1. Installasjon av XML::Parser

Forbered XML::Parser for kompilering:

```
perl Makefile.PL
```

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make test
```

Installer pakken:

```
make install
```

8.42.2. Innhold i XML::Parser

Installert modul: Expat.so

Korte beskrivelser

`Expat` gir Perl Expat grensesnittet

8.43. Intltool-0.51.0

Intltool er et internasjonaliseringsverktøy som brukes til å trekke ut oversettbare strenger fra kildefiler.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 1.5 MB

8.43.1. Installasjon av Intltool

Rett først en advarsel som er forårsaket av perl-5.22 og senere:

```
sed -i 's:\\\${:\\\$\\{:' intltool-update.in
```

Note

Det regulære uttrykket ovenfor ser uvanlig ut på grunn av alle skråstreker. Det den gjør er å legge til et skråstrek før høyre krøllparentes i sekvensen '\${' som resulterer i '\$\{'.

Forbered Intltool for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
install -v -Dm644 doc/I18N-HOWTO /usr/share/doc/intltool-0.51.0/I18N-HOWTO
```

8.43.2. Innhold i Intltool

Installerte programmer: intltool-extract, intltool-merge, intltool-prepare, intltool-update, og intltoolize
Installerte mapper: /usr/share/doc/intltool-0.51.0 og /usr/share/intltool

Korte beskrivelser

intltoolize Forbereder en pakke for å bruke intltool

intltool-extract Genererer deklarasjonsfiler som kan leses av **gettext**

intltool-merge Slår sammen oversatte strenger til forskjellige filtyper

intltool-prepare Oppdaterer pot filer og slår dem sammen med oversettelsesfiler

intltool-update Oppdaterer po malfilene og slår dem sammen med oversettelsene

8.44. Autoconf-2.71

Autoconf pakken inneholder programmer for å produsere skallskript som automatisk kan konfigurere kildekoden.

Omtrentlig byggetid: mindre enn 0.1 SBU (omtrent 6.8 SBU med testene)
Nødvendig diskplass: 24 MB

8.44.1. Installasjon av Autoconf

Forbered Autoconf for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Note

Testtiden for autoconf kan reduseres betydelig på en system med flere kjerner. For å gjøre dette, legg til **TESTSUITEFLAGS=-j<N>** til linjen over. For eksempel ved å bruke -j4 kan testtiden reduseres med over 60 prosent.

Installer pakken:

```
make install
```

8.44.2. Innhold i Autoconf

Installerte programmer: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, og ifnames
Installert mappe: /usr/share/autoconf

Korte beskrivelser

autoconf Produserer skallskript som automatisk konfigurerer programvares kildekodepakker for å tilpasse seg mange typer Unix-lignende systemer; konfigurasjonsskriptene den produserer er uavhengige— å kjøre de krever ikke **autoconf** programmet

autoheader Et verktøy for å lage malfiler av C *#define* uttrykk for configure å bruke

autom4te En innpakning for M4 makroprosessoren

autoreconf Kjører automatisk **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize**, og **libtoolize** i riktig rekkefølge for å spare tid når det gjøres endringer i **autoconf** og **automake** malfiler

autoscan Hjelper med å lage en `configure.in` fil for en programvarepakke; den undersøker kildefilene i et mappetreet, søker etter vanlige problemer med portabilitet, og oppretter en `configure.scan` fil som fungerer som en innledende `configure.in` fil for en pakke

autoupdate Endrer en `configure.in` fil som fortsatt anroper **autoconf** makroer ved deres gamle navn for å bruke gjeldende makronavn

ifnames Hjelper når det skrives `configure.in` filer for en programvarepakke; den skriver ut identifikatorene som pakken bruker i C forbehandler betingelser [Hvis en pakke allerede er satt for å ha en viss portabilitet, kan dette programmet hjelpe med å finne ut hva **configure** må sjekke etter. Den kan også fylle ut hull i en `configure.in` fil generert av **autoscan**.]

8.45. Automake-1.16.5

Automake pakken inneholder programmer for å generere Make filer for bruk med Autoconf.

Omtrentlig byggetid: mindre enn 0.1 SBU (omtrent 8.3 SBU med testene)
Nødvendig diskplass: 115 MB

8.45.1. Installasjon av Automake

Forbered Automake for kompilering:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.16.5
```

Kompiler pakken:

```
make
```

Å bruke make -j4 alternativet øker hastigheten på testene, selv på systemer med kun én prosessor, på grunn av interne forsinkelser i individuelle tester. Å teste resultatene, utsted:

```
make -j4 check
```

Testen t/subobj.sh er kjent for å mislykkes.

Installer pakken:

```
make install
```

8.45.2. Innholdet i Automake

Installerte programmer: aclocal, aclocal-1.16 (hardlinket til aclocal), automake, og automake-1.16 (hardlinket til automake)
Installerte mapper: /usr/share/aclocal-1.16, /usr/share/automake-1.16, og /usr/share/doc/automake-1.16.5

Korte beskrivelser

aclocal Genererer `aclocal.m4` filer basert på innholdet i `configure.in` filene

aclocal-1.16 En hard lenke til **aclocal**

automake Et verktøy for automatisk generering av `Makefile.in` filer fra `Makefile.am` filer [For å lage alle `Makefile.in` filer for en pakke, kjør dette programmet i mappen på øverste nivå. Ved å skanne `configure.in` filen, finner den automatisk hver passende `Makefile.am` fil og genererer tilsvarende `Makefile.in` fil.]

automake-1.16 En hard lenke til **automake**

8.46. OpenSSL-3.0.1

OpenSSL pakken inneholder administrasjonsverktøy og relaterte biblioteker til kryptografi. Disse er nyttige for å tilby kryptografiske funksjoner til andre pakker, for eksempel OpenSSH, e-postapplikasjoner og nettlesere (for tilgang til HTTPS-nettsteder).

Omtrentlig byggetid: 5.4 SBU
Nødvendig diskplass: 474 MB

8.46.1. Installasjon av OpenSSL

Forbered OpenSSL for kompilering:

```
./config --prefix=/usr         \
         --openssldir=/etc/ssl \
         --libdir=lib          \
         shared                \
         zlib-dynamic
```

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make test
```

En test, 30-test_afalg.t, er kjent for å mislykkes på noen kjerne konfigurasjoner (avhengig av inkonsistente verdier for CONFIG_CRYPTO_USER_API*-innstillinger.) Hvis det mislykkes, kan den trygt bli ignorert.

Installer pakken:

```
sed -i '/INSTALL_LIBS/s/libcrypto.a libssl.a//' Makefile
make MANSUFFIX=ssl install
```

Legg til versjonen i dokumentasjonskatalognavnet, for å være i samsvarer med andre pakker:

```
mv -v /usr/share/doc/openssl /usr/share/doc/openssl-3.0.1
```

Hvis ønskelig, installer litt tilleggsdokumentasjon:

```
cp -vfr doc/* /usr/share/doc/openssl-3.0.1
```

Note

Du bør oppdatere OpenSSL når en ny versjon som fikser sårbarheter er annonsert. Utgivelsene går i serier, med en bokstav for hver utgivelse etter den første utgivelsen (f.eks. 1.1.1, 1.1.1a, 1.1.1b, osv.). Fordi LFS installerer kun de delte bibliotekene, er det ikke nødvendig å rekompilere pakker som lenker til `libcrypto.so` eller `libssl.so` *ved oppgradering i samme serie.*

Imidlertid må alle kjørende programmer koblet til disse bibliotekene stoppes og startes på nytt. Les de relaterte oppføringene i Section 8.2.1, “Oppgraderingsproblemer” for detaljer.

8.46.2. Innhold i OpenSSL

Installerte programmer: c_rehash og openssl
Installerte biblioteker: libcrypto.so og libssl.so
Installerte mapper: /etc/ssl, /usr/include/openssl, /usr/lib/engines og /usr/share/doc/openssl-3.0.1

Korte beskrivelser

c_rehash er et Perl skript som skanner alle filer i en katalog og legger til symbolske lenker til hashverdiene deres

openssl er et kommandolinjeverktøy for bruk av de ulike kryptografifunksjonene til OpenSSL's kryptobibliotek fra skallet. Den kan brukes til ulike funksjoner som er dokumentert i **man 1 openssl**

`libcrypto.so` implementerer et bredt spekter av kryptografiske algoritmer som brukes i ulike Internett-standarder. Tjenestene som tilbys av dette biblioteket brukes av OpenSSL implementeringer av SSL, TLS og S/MIME, og de har også blitt brukt til å implementere OpenSSH, OpenPGP, og andre kryptografiske standarder

`libssl.so` implementerer protokollen Transport Layer Security (TLS v1). Det gir en rik API, dokumentasjon kan bli funnet ved å kjøre **man 3 ssl**

8.47. Kmod-29

Kmod pakken inneholder biblioteker og verktøy for lasting av kjerne moduler

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 12 MB

8.47.1. Installasjon av Kmod

Forbered Kmod for kompilering:

```
./configure --prefix=/usr           \
            --sysconfdir=/etc       \
            --with-openssl          \
            --with-xz               \
            --with-zstd             \
            --with-zlib
```

Betydningen av konfigureringsalternativene:

`--with-openssl`
Dette alternativet gjør det mulig for Kmod å håndtere PKCS7 signaturer for kjernemoduler.

`--with-xz`, *`--with-zlib`*, og *`--with-zstd`*
Disse alternativene gjør at Kmod kan håndtere komprimerte kjernemoduler.

Kompiler pakken:

```
make
```

Testpakken til denne pakken krever rå kjerneoverskrifter (ikke de “sanitiserte” kjernehodene installert tidligere), som er utenfor rammen av LFS.

Installer pakken og lag symbolkoblinger for kompatibilitet med Module-Init-Tools (pakken som tidligere håndterte Linux kjernemoduler):

```
make install

for target in depmod insmod modinfo modprobe rmmod; do
  ln -sfv ../bin/kmod /usr/sbin/$target
done

ln -sfv kmod /usr/bin/lsmod
```

8.47.2. Innhold i Kmod

Installerte programmer: depmod (lenker til kmod), insmod (lenker til kmod), kmod, lsmod (lenker til kmod), modinfo (lenker til kmod), modprobe (lenker til kmod), og rmmod (lenker til kmod)
Installert bibliotek: libkmod.so

Korte beskrivelser

depmod Oppretter en avhengighetsfil basert på symbolene den finner i eksisterende sett med moduler; denne avhengighetsfilen brukes av **modprobe** for automatisk å laste de nødvendige moduler

insmod	Installerer en lastbar modul i kjernen som kjører
kmod	Laster og laster ut kjernemoduler
lsmod	Viser innlastede moduler
modinfo	Undersøker en objektfil assosiert med en kjernemodul og viser all informasjon den kan hente
modprobe	Bruker en avhengighetsfil, opprettet av **depmod**, for automatisk å laste inn relevante moduler
rmmod	Laster ut moduler fra kjernen som kjører
`libkmod`	Dette biblioteket brukes av andre programmer til å laste inn og laste ut kjernemoduler

8.48. Libelf fra Elfutils-0.186

Libelf er et bibliotek for håndtering av ELF (kjørbare og linkbare formater) filer.

Omtrentlig byggetid: 0.9 SBU
Nødvendig diskplass: 116 MB

8.48.1. Installasjon av Libelf

Libelf er en del av elfutils-0.186 pakken. Bruk elfutils-0.186.tar.bz2 som kilde tarball.

Forbered Libelf for kompilering:

```
./configure --prefix=/usr                \
            --disable-debuginfod         \
            --enable-libdebuginfod=dummy
```

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer kun Libelf:

```
make -C libelf install
install -vm644 config/libelf.pc /usr/lib/pkgconfig
rm /usr/lib/libelf.a
```

8.48.2. Innhold i Libelf

Installert bibliotek: libelf.so (symlink) og libelf-0.186.so
Installert mappe: /usr/include/elfutils

Korte beskrivelser

`libelf` Inneholder API funksjoner for å håndtere ELF objektfiler

8.49. Libffi-3.4.2

Libffi-biblioteket gir et flyttbart programmeringsgrensesnitt på høyt nivå til ulike kallkonvensjoner. Dette lar en programmerer kalle enhver funksjon ved kjøring, spesifisert av en grensesnittbeskrivelse for et kall.

Omtrentlig byggetid: 1.9 SBU
Nødvendig diskplass: 10 MB

8.49.1. Installasjon av Libffi

Note

I likhet med GMP bygges libffi med spesifikke optimaliseringer til prosessoren som er i bruk. Hvis du bygger for et annet system, eksporter CFLAGS og CXXFLAGS for å spesifisere en generisk konstruksjon for arkitekturen din. Hvis dette ikke gjøres, vil alle applikasjoner som lenker til libffi utløse ulovlige operasjonsfeil.

Forbered libffi for kompilering:

```
./configure --prefix=/usr          \
            --disable-static       \
            --with-gcc-arch=native \
            --disable-exec-static-tramp
```

Betydningen av konfigureringsalternativet:

`--with-gcc-arch=native`
Sørger for at GCC optimerer for det gjeldende systemet. Hvis dette ikke er spesifisert, gjettes systemet og koden som genereres kanskje ikke er riktig for enkelte systemer. Hvis den genererte koden vil bli kopiert fra det opprinnelige systemet til et mindre kapabelt system system, bruk det mindre kapable systemet som parameter. For detaljer om alternative systemtyper, se *x86 alternativene i GCC manualen*.

`--disable-exec-static-tramp`
Deaktiver statisk trampolinestøtte. Det er en ny sikkerhet funksjon i libffi, men noen BLFS pakker (spesielt GJS og gobject-introspection) har ikke vært tilpasset det.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.49.2. Innhold i Libffi

Installert bibliotek: libffi.so

Korte beskrivelser

`libffi` inneholder API funksjonene for fremmede funksjonsgrensesnitt

8.50. Python-3.10.2

Python 3 pakken inneholder Python utviklingsmiljøet. Den er nyttig for objektorientert programmering, skriving av skript, prototyping store programmer, eller utvikle hele applikasjoner.

Omtrentlig byggetid: 4.3 SBU
Nødvendig diskplass: 275 MB

8.50.1. Installasjon av Python 3

Forbered Python for kompilering:

```
./configure --prefix=/usr        \
            --enable-shared      \
            --with-system-expat  \
            --with-system-ffi    \
            --with-ensurepip=yes \
            --enable-optimizations
```

Betydningen av konfigureringsalternativene:

`--with-system-expat`
Denne bryteren muliggjør kobling mot systemversjonen av Expat.

`--with-system-ffi`
Denne bryteren muliggjør kobling mot systemversjonen av libffi.

`--with-ensurepip=yes`
Denne bryteren gjør det mulig å bygge **pip** og **setuptools** pakkeprogrammer.

`--enable-optimizations`
Denne bryteren muliggjør stabile, men dyre, optimaliseringer.

Kompiler pakken:

```
make
```

Det anbefales ikke å kjøre testene på dette tidspunktet. Tester er kjent for å henge på ubestemt tid i det delvise LFS miljøet. Om ønskelig kan testene kjøres på nytt på slutten av dette kapittelet eller når Python 3 er reinstallert i BLFS. For å kjøre testene uansett, utsted **make test**.

Installer pakken:

```
make install
```

Hvis ønskelig, installer den forhåndsformaterte dokumentasjonen:

```
install -v -dm755 /usr/share/doc/python-3.10.2/html

tar --strip-components=1  \
    --no-same-owner       \
    --no-same-permissions \
    -C /usr/share/doc/python-3.10.2/html \
    -xvf ../python-3.10.2-docs-html.tar.bz2
```

Betydningen av dokumentasjons installasjons kommandoene:

`--no-same-owner` og `--no-same-permissions`
Sørg for at de installerte filene har riktig eierskap og tillatelser. Uten disse alternativene, å bruke tar vil installere pakkefilene med oppstrøms skaperens verdier.

8.50.2. Innhold i Python 3

Installerte programmer: 2to3, idle3, pip3, pydoc3, python3, og python3-config
Installert bibliotek: libpython3.10.so og libpython3.so
Installerte mapper: /usr/include/python3.10, /usr/lib/python3, og /usr/share/doc/python-3.10.2

Korte beskrivelser

2to3 er et Python program som leser Python 2.x kildekoden og bruker en serie reparasjoner for å forvandle den til gyldig Python 3.x kode

idle3 er et innpakningsskript som åpner en Python bevisst GUI tekstprogram. For at dette skriptet skal kjøre, må du ha installert Tk før Python slik at Tkinter Python modulen blir bygget

pip3 Pakkeinstallasjonsprogrammet for Python. Du kan bruke pip til å installere pakker fra Python Pakke Indeks og andre indekser

pydoc3 er Python dokumentasjonsverktøy

python3 er en tolket, interaktiv, objektorientert programmerings språk

8.51. Ninja-1.10.2

Ninja er et lite byggesystem med fokus på hastighet.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 64 MB

8.51.1. Installasjon av Ninja

Når den kjøres, kjører ninja normalt et maksimalt antall prosesser parallelt. Som standard er dette antall kjerner på systemet pluss to. I noen tilfeller kan dette overopphete en CPU eller kjøre et system ut av minne. Hvis du kjører fra kommandolinjen, sender du en -jN-parameter vil det begrense antall parallelle prosesser, men noen pakker bygger inn utførelsen av ninja og sender ikke en -j parameter.

Ved å bruke *optional* prosedyren nedenfor lar en bruker begrense antall parallelle prosesser via en miljøvariabel, NINJAJOBS. **For eksempel**, å sette:

```
export NINJAJOBS=4
```

vil begrense ninja til fire parallelle prosesser.

Hvis ønskelig, legg til muligheten til å bruke miljøvariabelen NINJAJOBS ved å kjøre:

```
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
```

Bygg Ninja med:

```
python3 configure.py --bootstrap
```

Betydningen av byggealternativet:

`--bootstrap`
Denne parameteren tvinger ninja til å gjenoppbygge seg selv for gjeldene system.

For å teste resultatene, utsted:

```
./ninja ninja_test
./ninja_test --gtest_filter=-SubprocessTest.SetWithLots
```

Installer pakken:

```
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
```

8.51.2. Innhold av Ninjainja

Installerte programmer: ninja

Korte beskrivelser

ninja er Ninja byggesystemet

8.52. Meson-0.61.1

Meson er et åpen kildekode byggesystem ment å være både ekstremt raskt og så brukervennlig som mulig.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 41 MB

8.52.1. Installasjon av Meson

Kompiler Meson med følgende kommando:

```
python3 setup.py build
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
python3 setup.py install --root=dest
cp -rv dest/* /
install -vDm644 data/shell-completions/bash/meson /usr/share/bash-completion/compl
install -vDm644 data/shell-completions/zsh/_meson /usr/share/zsh/site-functions/_m
```

Betydningen av installasjonsparametrene:

`--root=dest`
Som standard **python3 setup.py install** installerer ulike filer (som man-sider) i Python Eggs. Med en spesifisert rotplassering, **setup.py** installerer disse filene inn i et standard hierarki. Så hierarkiet kan bare kopieres til standardplasseringen.

8.52.2. Innhold i Meson

Installerte programmer: meson
Installert mappe: /usr/lib/python3.10/site-packages/meson-0.61.1-py3.10.egg-info and /usr/lib/python3.10/site-packages/mesonbuild

Korte beskrivelser

meson Et byggesystem med høy produktivitet

8.53. Coreutils-9.0

Pakken Coreutils inneholder verktøy for å vise og stille inn grunnleggende systemegenskaper.

Omtrentlig byggetid: 2.6 SBU
Nødvendig diskplass: 153 MB

8.53.1. Installasjon av Coreutils

POSIX krever at programmer fra Coreutils gjenkjenner karakter grenser riktig selv i multibyte lokaliteter. Følgende oppdateringer fikser dette misligholdet og andre internasjonaliseringsrelaterte feil.

```
patch -Np1 -i ../coreutils-9.0-i18n-1.patch
```

Note

Tidligere ble det funnet mange feil i denne oppdateringen. Ved melding om nye feil til Coreutils vedlikeholdere, vennligst først sjekk om de er reproduserbare uten denne oppdateringen.

Nå, fiks et problem med chmod returverdier:

```
patch -Np1 -i ../coreutils-9.0-chmod_fix-1.patch
```

Forbered nå Coreutils for kompilering:

```
autoreconf -fiv
FORCE_UNSAFE_CONFIGURE=1 ./configure \
            --prefix=/usr            \
            --enable-no-install-program=kill,uptime
```

Betydningen av konfigureringsalternativene:

autoreconf
Oppdateringen for internasjonalisering har modifisert byggesystemet til pakken, slik at konfigurasjonsfilene må bli regenerert.

`FORCE_UNSAFE_CONFIGURE=1`
Denne miljøvariabelen lar pakken bli bygget som `root` brukeren.

`--enable-no-install-program=kill,uptime`
Hensikten med denne bryteren er å hindre Coreutils fra installere binærfiler som vil bli installert av andre pakker senere.

Kompiler pakken:

```
make
```

Hopp ned til “Installer pakken” hvis du ikke kjører testpakken.

Nå er testpakken klar til å kjøres. Kjør først testene som er ment å kjøres som bruker `root`:

```
make NON_ROOT_USERNAME=tester check-root
```

Vi kommer til å kjøre resten av testene som brukeren `tester`. Visse tester krever at brukeren er medlem av mer enn én gruppe. Sånn at disse testene ikke hoppes over, legg til en midlertidig gruppe og gjør bruker `tester` en del av det:

```
echo "dummy:x:102:tester" >> /etc/group
```

Fiks noen av tillatelsene slik at ikke-root brukeren kan kompilere og kjøre testene:

```
chown -Rv tester .
```

Kjør nå testene:

```
su tester -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes check"
```

Test-getlogin testen er kjent for å mislykkes i LFS chroot-miljøet.

Fjern den midlertidige gruppen:

```
sed -i '/dummy/d' /etc/group
```

Installer pakken:

```
make install
```

Flytt programmer til stedene spesifisert av 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
```

8.53.2. Innhold i Coreutils

Installerte programmer: [, 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, og yes
Installert bibliotek: libstdbuf.so (i /usr/libexec/coreutils)
Installert mappe: /usr/libexec/coreutils

Korte beskrivelser

[Er faktisk en kommando, /usr/bin/[, det er et synonym for **test** kommandoen

base32 Koder og dekoder data i henhold til base32 spesifikasjonen (RFC 4648)

base64 Koder og dekoder data i henhold til base64 spesifikasjonen (RFC 4648)

b2sum Skriver ut eller kontrollerer BLAKE2 (512-bit) sjekksummer

basename Fjerner enhver bane og et gitt suffiks fra et filnavn

basenc Koder eller dekoder data ved hjelp av ulike algoritmer

cat Slår sammen filer til standard utdata

chcon Endrer sikkerhetskontekst for filer og mapper

chgrp Endrer gruppeeierskap for filer og mapper

chmod Endrer tillatelsene til hver fil til gitt modus; modusen kan enten være en symbolsk representasjon av endringene som skal gjøres eller en oktalt tall som representerer de nye tillatelsene

chown Endrer bruker- og/eller gruppeeierskap av filer og kataloger

chroot Kjører en kommando med den angitte mappen som / mappe

cksum Skriver ut sjekksummen for syklisk redundanssjekk (CRC) og antall byte for hver spesifisert fil

comm Sammenligner to sorterte filer, og skriver ut i tre kolonner, linjene som er unike og linjene som er vanlige

cp Kopierer filer

csplit Deler en gitt fil i flere nye filer, og skiller dem i henhold til gitte mønstre eller linjenummer og skriver ut antall byte av hver nye fil

cut Skriver ut seksjoner av linjer, og velger delene i henhold til gitte felt eller posisjoner

date Viser gjeldende tid i det gitte formatet, eller stiller inn systemdato

dd Kopierer en fil med den gitte blokkstørrelsen og antallet, mens det valgfritt utføres konverteringer på den

df Rapporterer hvor mye diskplass som er tilgjengelig (og brukt) på alle monterte filsystemer, eller bare på filsystemene som inneholder de valgte filer

dir Viser innholdet i hver gitt katalog (det samme som **ls** kommandoen)

dircolors Skriver ut kommandoer for å angi LS_COLOR miljøvariabel for å endre fargeskjemaet som brukes av **ls**

dirname Fjerner ikke-mappesuffikset fra et filnavn

du Rapporterer hvor mye diskplass som brukes av gjeldende katalog, av hver av de gitte katalogene (inkludert alle underkataloger) eller av hver av de gitte filene

echo Viser de gitte strengene

env Kjører en kommando i et modifisert miljø

expand Konverterer tabulatorer til mellomrom

expr Evaluerer uttrykk

factor Skriver ut primfaktorene til alle spesifiserte heltall

false Gjør ingenting, mislykket; den avsluttes alltid med en statuskode som indikerer feil

fmt Reformaterer avsnittene i de gitte filene

fold Omslutter linjene i de gitte filene

groups Rapporterer en brukers gruppemedlemskap

head Skriver ut de ti første linjene (eller gitt antall linjer) av hver gitt fil

hostid Rapporterer den numeriske identifikatoren (i heksadesimal) til verten

id Rapporterer effektiv brukerID, gruppeID og gruppemedlemskap av gjeldende bruker eller en spesifisert bruker

install Kopierer filer mens de angir tillatelsesmoduser og, hvis mulig, deres eier og gruppe

join Kobler sammen linjene som har identiske sammenføyningsfelt fra to separate filer

link	Oppretter en hard link med det gitte navnet til en fil
ln	Lager harde koblinger eller myke (symbolske) koblinger mellom filer
logname	Rapporterer gjeldende brukers påloggingsnavn
ls	Viser innholdet i hver gitt katalog
md5sum	Rapporterer eller kontrollerer Message Digest 5 (MD5) sjekksummer
mkdir	Oppretter en mappe med gitt navn
mkfifo	Oppretter først inn, først ut (FIFOs), en "navngitt kanal (pipe)" på UNIX-språk, med gitt navn
mknod	Oppretter enhetsnoder med de gitte navnene; en enhetsnode er en spesialfil for tegn, en spesialfil for blokk eller en FIFO
mktemp	Oppretter midlertidige filer på en sikker måte; det brukes i skript
mv	Flytter eller gir nytt navn til filer eller kataloger
nice	Kjører et program med endret skjemaprioritet
nl	Nummerer linjene fra de gitte filene
nohup	Kjører en kommando som er immun mot avbrudd, med utdata omdirigert til en loggfil
nproc	Skriver ut antall tilgjengelige prosesseringsenheter for en prosess
numfmt	Konverterer tall til eller fra menneskelesbare strenger
od	Dumper filer i oktale og andre formater
paste	Slår sammen de gitte filene og kobler sammen sekvensielt tilsvarende linjer side ved side, atskilt med tabulatortegn
pathchk	Sjekker om filnavn er gyldige eller flyttbare
pinky	Er en lettvekts fingerklient; den rapporterer noe informasjon om de gitte brukerne
pr	Paginerer og spalter filer for utskrift
printenv	Skriver ut miljøet
printf	Skriver ut de gitte argumentene i henhold til det gitte formatet, mye som C printf funksjonen
ptx	Produserer en permutert indeks fra innholdet i de gitte filene, med hvert søkeord i sin kontekst
pwd	Rapporterer navnet på gjeldende arbeidskatalog
readlink	Rapporterer verdien av den gitte symbolske lenken
realpath	Skriver ut den løste banen
rm	Fjerner filer eller mapper
rmdir	Fjerner mapper hvis de er tomme
runcon	Kjører en kommando med spesifisert sikkerhetskontekst
seq	Skriver ut en sekvens av tall innenfor et gitt område og med en gitt økning
sha1sum	Skriver ut eller sjekker 160-bits Secure Hash Algorithm 1 (SHA1) sjekksummer
sha224sum	Skriver ut eller kontrollerer 224-biters Secure Hash Algoritme sjekksummer
sha256sum	Skriver ut eller kontrollerer 256-biters Secure Hash Algoritme sjekksummer
sha384sum	Skriver ut eller kontrollerer 384-biters Secure Hash Algoritme sjekksummer

sha512sum Skriver ut eller kontrollerer 512-biters Secure Hash Algoritme sjekksummer

shred Overskriver de gitte filene gjentatte ganger med komplekse mønstre, som gjør det vanskelig å gjenopprette dataene

shuf Blander tekstlinjer

sleep Pauser i den gitte tiden

sort Sorterer linjene fra de gitte filene

split Deler den gitte filen i biter, etter størrelse eller antall linjer

stat Viser fil- eller filsystemstatus

stdbuf Kjører kommandoer med endrede bufferoperasjoner for standard dataflyt

stty Angir eller rapporterer terminallinjeinnstillinger

sum Skriver ut sjekksum og blokktellinger for hver gitt fil

sync Tømmer filsystembuffere; den tvinger endrede blokker til disk og oppdaterer superblokken

tac Sammenslår de gitte filene i revers

tail Skriver ut de ti siste linjene (eller gitt antall linjer) av hver gitt fil

tee Leser fra standard inngang mens du skriver både til standard utgang og til de gitte filene

test Sammenligner verdier og kontrollerer filtyper

timeout Kjører en kommando med en tidsbegrensning

touch Endrer filtidsstempler, angir tilgang og endrings tider for de gitte filene til gjeldende tid; filer som ikke eksisterer opprettes med null lengde

tr Oversetter, klemmer sammen og sletter de gitte tegnene fra standard inngang

true Gjør ingenting, vellykket; den avsluttes alltid med en statuskode som indikerer suksess

truncate Krymper eller utvider en fil til den angitte størrelsen

tsort Utfører en topologisk sortering; den skriver en fullstendig ordnet liste i henhold til delbestillingen i en gitt fil

tty Rapporterer filnavnet til terminalen som er koblet til standard inngang

uname Rapporterer systeminformasjon

unexpand Konverterer mellomrom til tabulatorer

uniq Forkaster alle unntatt én av påfølgende identiske linjer

unlink Fjerner den gitte filen

users Rapporterer navnene på brukerne som er logget på

vdir Er det samme som **ls -l**

wc Rapporterer antall linjer, ord og byte for hver gitt fil, samt en totale linjer når mer enn én fil er gitt

who Rapporterer hvem som er pålogget

whoami Rapporterer brukernavnet som er knyttet til gjeldende effektive brukerID

yes Skriver ut “y” gjentatte ganger eller en gitt streng til den drepes

`libstdbuf` Bibliotek brukt av **stdbuf**

8.54. Check-0.15.2

Check er et rammeverk for C enhetstesting.

Omtrentlig byggetid: 0.1 SBU (omtrent 3.8 SBU med testene)
Nødvendig diskplass: 12 MB

8.54.1. Installasjon av Check

Forbered sjekk for kompilering:

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

Bygg pakken:

```
make
```

Samlingen er nå fullført. For å kjøre testpakkene for Check, utsted følgende kommando:

```
make check
```

Installer pakken:

```
make docdir=/usr/share/doc/check-0.15.2 install
```

8.54.2. Innholdet i Check

Installert program: checkmk
Installert bibliotek: libcheck.so

Korte beskrivelser

checkmk	Awk skript for å generere C enhetstester for bruk med Check's rammeverk for enhetstesting
`libcheck.{a,so}`	Inneholder funksjoner som gjør at Check kan kalles fra et test program

8.55. Diffutils-3.8

Diffutils pakken inneholder programmer som viser forskjellene mellom filer eller mapper.

Omtrentlig byggetid: 0.6 SBU
Nødvendig diskplass: 34 MB

8.55.1. Installasjon av Diffutils

Forbered Diffutils for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.55.2. Innhold i Diffutils

Installerte programmer: cmp, diff, diff3, og sdiff

Korte beskrivelser

cmp Sammenligner to filer og rapporterer om eller i hvilken byte de avviker

diff Sammenligner to filer eller mapper og rapporterer hvilke linjer i filene som er forskjellige

diff3 Sammenligner tre filer linje for linje

sdiff Slår sammen to filer og viser resultatene interaktivt

8.56. Gawk-5.1.1

Gawk pakken inneholder programmer for å manipulere tekstfiler.

Omtrentlig byggetid: 0.4 SBU
Nødvendig diskplass: 43 MB

8.56.1. Installasjon av Gawk

Først, sørg for at noen unødvendige filer ikke blir installert:

```
sed -i 's/extras//' Makefile.in
```

Forbered Gawk for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

Hvis ønskelig, installer dokumentasjonen:

```
mkdir -pv                                   /usr/share/doc/gawk-5.1.1
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-5.1.1
```

8.56.2. Innhold i Gawk

Installerte programmer: awk (lenker til gawk), gawk, og awk-5.1.1
Installerte biblioteker: filefuncs.so, fnmatch.so, fork.so, inplace.so, intdiv.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoway.so, rwarray.so, og time.so (alle i /usr/lib/gawk)
Installerte mapper: /usr/lib/gawk, /usr/libexec/awk, /usr/share/awk, og /usr/share/doc/gawk-5.1.1

Korte beskrivelser

awk En lenke til **gawk**

gawk Et program for å manipulere tekstfiler; det er GNU implementeringen av **awk**

gawk-5.1.1 En hard lenke til **gawk**

8.57. Findutils-4.9.0

Findutils pakken inneholder programmer for å finne filer. Disse programmene er gitt for å rekursivt søke gjennom et katalogtre og til å opprette, vedlikeholde og søke i en database (ofte raskere enn den rekursive letingen, men er upålitelig hvis databasen ikke nylig har blitt oppdatert).

Omtrentlig byggetid: 0.9 SBU
Nødvendig diskplass: 51 MB

8.57.1. Installasjon av Findutils

Forbered Findutils for kompilering:

```
case $(uname -m) in
    i?86)   TIME_T_32_BIT_OK=yes ./configure --prefix=/usr --localstatedir=/var/li
    x86_64) ./configure --prefix=/usr --localstatedir=/var/lib/locate ;;
esac
```

Betydningen av konfigureringsalternativene:

TIME_32_BIT_OK=yes
Denne innstillingen er nødvendig for å bygge et 32 bit system.

`--localstatedir`
Dette alternativet endrer plasseringen av **locate** databasen til å være i `/var/lib/locate`, som er FHS kompatibel.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
chown -Rv tester .
su tester -c "PATH=$PATH make check"
```

Installer pakken:

```
make install
```

8.57.2. Innhold i Findutils

Installerte programmer: find, locate, updatedb, og xargs
Installert mappe: /var/lib/locate

Korte beskrivelser

find Søker i gitte katalogtrær etter filer som samsvarer med de spesifiserte kriterier

locate Søker gjennom en database med filnavn og rapporterer navnene som inneholder en gitt streng eller samsvarer med et gitt mønster

updatedb Oppdaterer **locate** databasen; den skanner hele filsystemet (inkludert andre filsystemer som for øyeblikket er montert, med mindre den blir bedt om å ikke gjøre det) og legger inn hvert filnavn den finner i databasen

xargs Kan brukes til å gi en gitt kommando til en liste over filer

8.58. Groff-1.22.4

Groff pakken inneholder programmer for prosessering og formatering av tekst.

Omtrentlig byggetid: 0.5 SBU
Nødvendig diskplass: 88 MB

8.58.1. Installasjon av Groff

Groff forventer miljøvariabelen PAGE å inneholde standard papirstørrelse. For brukere i USA, *PAGE=letter* er passende. Andre steder, *PAGE=A4* kan være mer egnet. Mens standard papirstørrelsen konfigureres under kompilering, kan den overstyres senere ved å sende enten "A4" eller "letter" til /etc/papersize filen.

Forbered Groff for kompilering:

```
PAGE=<paper_size> ./configure --prefix=/usr
```

Denne pakken støtter ikke parallellbygging. Kompiler pakken:

```
make -j1
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make install
```

8.58.2. Innhold i Groff

Installerte programmer: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, glilypond, gperl, gpinyin, grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, gropdf, grops, grotty, hpftodit, 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, og troff
Installerte mapper: /usr/lib/groff og /usr/share/doc/groff-1.22.4, /usr/share/groff

Korte beskrivelser

addftinfo Leser en troff fontfil og legger til noen ekstra fontmetrikk informasjon som brukes av **groff** systemet

afmtodit Oppretter en fontfil for bruk med **groff** og **grops**

chem Groff forprosessor for å lage kjemiske strukturdiagrammer

eqn Kompilerer beskrivelser av ligninger innebygd i troff inndata filer i kommandoer som forstås av **troff**

eqn2graph Konverterer en troff EQN (ligning) til et beskåret bilde

gdiffmk Markerer forskjeller mellom groff/nroff/troff filer

glilypond Forvandler noter skrevet på lilypond språket til groff språket

gperl Forprosessor for groff, tillater tillegg av perl kode inn i groff filer

gpinyin Forbehandler for groff, tillater tillegg av kinesisk Europeisk lignende språk pinyin til groff filer.

grap2graph Konverterer et grafdiagram til et beskåret punktgrafikkbilde

grn	En **groff** forbehandler for gremlin filer
grodvi	En driver for **groff** som produserer TeX dvi format
groff	En frontend til groff dokumentformateringssystem; normalt, kjøres **troff** program og en post-prosessor passende for den valgte enheten
groffer	Viser groff filer og man sider på X og tty terminaler
grog	Leser filer og gjetter hvilke av **groff** alternativer `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s`, og `-t` kreves for å skrive ut filer, og rapporterer **groff** kommandoen inkludert disse alternativene
grolbp	Er en **groff** driver for Canon CAPSL skrivere (laserskrivere i LBP-4 og LBP-8 serien)
grolj4	Er en driver for **groff** som produserer utdata i PCL5 formatet som passer for en HP LaserJet 4 skriver
gropdf	Oversetter utdataene for GNU **troff** til PDF
grops	Oversetter utdataene for GNU **troff** til PostScript
grotty	Oversetter utdataene for GNU **troff** inn i en form som passer for skrivemaskinlignende enheter
hpftodit	Oppretter en fontfil for bruk med **groff -Tlj4** fra en HP merket font metrisk fil
indxbib	Oppretter en invertert indeks for de bibliografiske databasene med en spesifisert fil for bruk med **refer**, **lookbib**, og **lkbib**
lkbib	Søker i bibliografiske databaser etter referanser som inneholder spesifiserte nøkler og rapporterer eventuelle referanser som er funnet
lookbib	Skriver ut en melding om standardfeil (med mindre standardinndata). ikke er en terminal), leser en linje som inneholder et sett med nøkkelord fra standard inndata, søker i de bibliografiske databasene i en spesifisert fil for referanser som inneholder disse nøkkelordene, skriver da ut eventuelle referanser som er funnet på standardutgangen, og gjentar denne prosessen til slutten av inndataen
mmroff	En enkel forprosessor for **groff**
neqn	Formaterer utdata ligninger for amerikansk standardkode for informasjon utveksling (ASCII)
nroff	Et skript som emulerer **nroff** kommandoen ved hjelp av **groff**
pdfmom	Er en innpakning rundt groff som letter produksjonen av PDF dokumenter fra filer formatert med mom makroene.
pdfroff	Oppretter pdf dokumenter ved hjelp av groff
pfbtops	Oversetter en PostScript font i `.pfb` formatet til ASCII
pic	Kompilerer beskrivelser av bilder innebygd i troff eller TeX inndatafiler til kommandoer som forstås av TeX eller **troff**
pic2graph	Konverterer et PIC diagram til et beskåret bilde
post-grohtml	Oversetter utdataene til GNU **troff** til HTML
preconv	Konverterer koding av inndatafiler til noe GNU **troff** forstår
pre-grohtml	Oversetter utdataene til GNU **troff** til HTML
refer	Kopierer innholdet i en fil til standardutgang, unntatt de som går mellom *.[* og *.]* tolkes som referanser, og linjer mellom *.R1* og *.R2* tolkes som kommandoer for hvordan referanser skal behandles

roff2dvi	Transformerer Roff filer til DVI format
roff2html	Transformerer Roff filer til HTML format
roff2pdf	Transformerer Roff filer til PDFer
roff2ps	Transformerer Roff filer til ps filer
roff2text	Transformerer Roff filer til text filer
roff2x	Transformerer Roff filer til andre formater
soelim	Leser filer og erstatter linjer i formatet *.so filer* av innholdet i nevnte *filer*
tbl	Kompilerer beskrivelser av tabeller innebygd i troff inndata filer til kommandoer som forstås av **troff**
tfmtodit	Oppretter en fontfil for bruk med **groff -Tdvi**
troff	Er veldig kompatibel med Unix **troff**; den bør vanligvis startes ved hjelp av **groff** kommandoen, som også vil kjøre pre- og post-prosessorer i riktig rekkefølge og med passende alternativer

8.59. GRUB-2.06

GRUB pakken inneholder en oppstartslaster (GRand Unified Bootloader).

Omtrentlig byggetid: 0.7 SBU
Nødvendig diskplass: 158 MB

8.59.1. Installasjon av GRUB

Note

Hvis systemet ditt har UEFI støtte og du ønsker å starte LFS med UEFI, kan du hoppe over denne pakken i LFS, og installere GRUB med UEFI støtte (og dets avhengigheter) som følger *BLFS siden* på slutten av dette kapittelet.

Forbered GRUB for kompilering:

```
./configure --prefix=/usr          \
            --sysconfdir=/etc      \
            --disable-efiemu       \
            --disable-werror
```

Betydningen av de nye konfigureringsalternativene:

`--disable-werror`
Dette gjør at bygget kan fullføres med advarsler fra nyere Flex versjoner.

`--disable-efiemu`
Dette alternativet minimerer det som bygges ved å deaktivere en funksjon og testprogrammer som ikke er nødvendig for LFS.

Kompiler pakken:

```
make
```

Testpakken for denne pakken anbefales ikke. Mesteparten av testene avhenger av pakker som ikke er tilgjengelige i det begrensete LFS miljøet. For å kjøre testene uansett, kjør **make check**.

Installer pakken:

```
make install
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

Bruk av GRUB for å gjøre LFS systemet oppstartbart vil bli diskutert i Section 10.4, “Bruke GRUB til å sette opp oppstartsprosessen”.

8.59.2. Innhold i GRUB

Installerte programmer: grub-bios-setup, grub-editenv, grub-file, grub-fstest, grub-glue-efi, grub-install, grub-kbdcomp, grub-macbless, grub-menulst2cfg, 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, og grub-syslinux2cfg
Installerte mapper: /usr/lib/grub, /etc/grub.d, /usr/share/grub, og /boot/grub (når grub-install kjøres for første gang)

Korte beskrivelser

grub-bios-setup Er et hjelpeprogram for grub-install

grub-editenv Et verktøy for å redigere miljøblokken

grub-file Sjekker om FILE er av den angitte typen.

grub-fstest Verktøy for å feilsøke filsystemdriveren

grub-glue-efi Behandler ia32 og amd64 EFI bilder og limer dem i henhold til Apple-format.

grub-install Installer GRUB på harddisken din

grub-kbdcomp Skript som konverterer et xkb oppsett til et som gjenkjennes av GRUB

grub-macbless Velsignelse i Mac-stil på HFS eller HFS+ filer

grub-menulst2cfg Konverterer en GRUB Legacy `menu.lst` til en `grub.cfg` for bruk med GRUB 2

grub-mkconfig Generer en grub konfigurasjonsfil

grub-mkimage Lager et oppstartbart bilde av GRUB

grub-mklayout Genererer en GRUB tastaturoppsettfil

grub-mknetdir Forbereder en GRUB netboot mappe

grub-mkpasswd-pbkdf2 Genererer et kryptert PBKDF2 passord for bruk i oppstartsmenyen

grub-mkrelpath Gir et systembanenavn i forhold til roten

grub-mkrescue Lager et oppstartbart bilde av GRUB som passer for en diskett eller CDROM/DVD

grub-mkstandalone Genererer et frittstående bilde

grub-ofpathname Er et hjelpeprogram som skriver ut banen til en GRUBenhet

grub-probe Undersøk enhetsinformasjon for en gitt bane eller enhet

grub-reboot Angir standard oppstartsoppføring for GRUB bare for neste oppstart

grub-render-label Gjengi Apple .disk_label for Apple Mac-er

grub-script-check Sjekker GRUB konfigurasjonsskriptet for syntaksfeil

grub-set-default Angir standard oppstartsoppføring for GRUB

grub-sparc64-setup Er et hjelpeprogram for grub-setup

grub-syslinux2cfg Forvandler en syslinux konfigurasjonsfil til grub.cfg format

8.60. Gzip-1.11

Gzip pakken inneholder programmer for komprimering og dekomprimering av filer.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 20 MB

8.60.1. Installasjon av Gzip

Forbered Gzip for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.60.2. Contents of Gzip

Installerte programmer: gunzip, gzexe, gzip, uncompress (hard lenket med gunzip), zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore, og znew

Korte beskrivelser

gunzip Dekomprimerer gzippede filer

gzexe Oppretter selvdekomprimerende kjørbare filer

gzip Komprimerer de gitte filene ved å bruke Lempel-Ziv (LZ77) koding

uncompress Dekomprimerer komprimerte filer

zcat Dekomprimerer de gitte gzip filene til standard utgang

zcmp Kjører **cmp** på gzippede filer

zdiff Kjører **diff** på gzippede filer

zegrep Kjører **egrep** på gzippede filer

zfgrep Kjører **fgrep** på gzippede filer

zforce Tvinger et `.gz` filetternavn på alle gitte filer som er gzippede filer, slik at **gzip** ikke vil komprimere dem igjen; dette kan være nyttig når filnavn ble avkortet under en filoverføring

zgrep Kjører **grep** på gzippede filer

zless Kjører **less** på gzippede filer

zmore Kjører **more** på gzippede filer

znew Re-komprimerer filer fra **compress** format til **gzip** format—`.Z` til `.gz`

8.61. IPRoute2-5.16.0

IPRoute2 pakken inneholder programmer for grunnleggende og avansert IPV4 basert nettverk.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 15 MB

8.61.1. Installasjon av IPRoute2

arpd programmet inkludert i denne pakken vil ikke bygges siden den er avhengig av Berkeley DB, som ikke er installert i LFS. Men en mappe for **arpd** og en man side vil fortsatt bli installert. Forhindre dette ved å kjøre kommandoene nedenfor. Hvis **arpd** binær er nødvendig, instruksjoner for kompilering av Berkeley DB finnes i BLFS-boken på *https://www.linuxfromscratch.org/blfs/view/stable-systemd/server/db.html*.

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

Kompiler pakken:

```
make
```

Denne pakken har ikke en fungerende testpakke.

Installer pakken:

```
make SBINDIR=/usr/sbin install
```

Hvis ønskelig, installer dokumentasjonen:

```
mkdir -pv             /usr/share/doc/iproute2-5.16.0
cp -v COPYING README* /usr/share/doc/iproute2-5.16.0
```

8.61.2. Innhold i IPRoute2

Installerte programmer: bridge, ctstat (lenker til lnstat), genl, ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (lenker til lnstat), ss, og tc
Installerte mapper: /etc/iproute2, /usr/lib/tc, og /usr/share/doc/iproute2-5.16.0

Korte beskrivelser

bridge Konfigurerer nettverksbroer

ctstat Verktøy for tilkoblingsstatus

genl Generisk verktøy for netlink grenseflate

ifcfg Et skall skriptinnpakning for **ip** kommando [Merk at det krever **arping** og **rdisk** programmer fra iputils pakken som finnes på *http://www.skbuff.net/iputils/*.]

ifstat Viser grensesnittstatistikken, inkludert mengden av overførte og mottatte pakker via et grensesnitt

ip Den viktigste kjørbare. Den har flere forskjellige funksjoner:
ip link *<device>* lar brukere se på enhetens tilstand og gjøre endringer
ip addr lar brukere se på adresser og egenskapene deres, legge til nye adresser og slette gamle
ip neighbor lar brukerne se på nabo bindinger og deres egenskaper, legge til nye nabooppføringer og slette gamle

ip rule allows users to look at the routing policies and change them
ip route lar brukerne se på rutingtabellen og endre rutetabellregler
ip tunnel lar brukere se på IP tunneler og deres egenskaper, og endre dem
ip maddr lar brukerne se på multicast adresser og deres egenskaper, og endre dem
ip mroute allows users to set, change, or delete the multicast routing
ip monitor lar brukerne overvåke kontinuerlig tilstanden til enheter, adresser og ruter

lnstat Gir Linux nettverksstatistikk; det er en generalisert og mer funksjonsfull erstatning for det gamle **rtstat** programmet

nstat Viser nettverksstatistikk

routef En komponent av **ip route**. Dette er for å tømme rutetabellene

routel En komponent av **ip route**. Dette er for liste rutetabellene

rtacct Viser innholdet i `/proc/net/rt_acct`

rtmon Overvåkingsverktøy for Route

rtpr Konverterer utdataene til **ip -o** tilbake til en lesbar form

rtstat Statusverktøy for Route

ss Ligner på **netstat** kommandoen; viser aktive forbindelser

tc Trafikkkontrollerende kjørbar; dette er for Quality Of Service (QOS) og Class Of Service (COS) implementeringer
tc qdisc lar brukere sette opp køen discipline
tc class lar brukere sette opp klasser basert på køen til discipline planlegging
tc estimator lar brukerne estimere nettverksflyt inn i et nettverk
tc filter lar brukere sette opp QOS/COS pakkefiltrering
tc policy lar brukere sette opp QOS/COS retningslinjer

8.62. Kbd-2.4.0

Kbd pakken inneholder nøkkeltabellfiler, konsollfonter og tastatur verktøy.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 33 MB

8.62.1. Installasjon av Kbd

Oppførselen til tilbaketastene og slettetastene er ikke konsistent på tvers av nøkkelkartene i Kbd pakken. Følgende oppdatering fikser dette problem for i386 keymaps:

```
patch -Np1 -i ../kbd-2.4.0-backspace-1.patch
```

Etter oppdateringen, genererer tilbaketasten tegnet med kode 127, og slettetasten genererer en velkjent escape sekvens.

Fjern det overflødige **resizecons** programmet (det krever den nedlagte svgalib for å gi videomodusfilene - for normal bruk **setfont** gjør størrelsen på konsollen passende) sammen med dens Manside.

```
sed -i '/RESIZECONS_PROGS=/s/yes/no/' configure
sed -i 's/resizecons.8 //' docs/man/man8/Makefile.in
```

Forbered Kbd for kompilering:

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

Betydningen av konfigureringsalternativet:

`--disable-vlock`
Dette alternativet forhindrer at vlock verktøyet blir bygget fordi det krever PAM biblioteket, som ikke er tilgjengelig i chroot miljøet.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

Note

For noen språk (f.eks. hviterussisk) gir ikke Kbd pakken et nyttig nøkkelkart der standard keymap antar ISO-8859-5-kodingen og CP1251-tastaturet brukes vanligvis. Brukere av slike språk må laste ned fungerende keymaps separat.

Hvis ønskelig, installer dokumentasjonen:

```
mkdir -pv           /usr/share/doc/kbd-2.4.0
cp -R -v docs/doc/* /usr/share/doc/kbd-2.4.0
```

8.62.2. Innhold i Kbd

Installerte programmer: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link to psfxtable), psfgettable (link to psfxtable), psfstriptable (link to psfxtable), psfxtable, setfont, setkeycodes, setleds, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start, og unicode_stop

Installerte mapper: /usr/share/consolefonts, /usr/share/consoletrans, /usr/share/keymaps, /usr/share/doc/kbd-2.4.0, og /usr/share/unimaps

Korte beskrivelser

chvt Endrer den virtuelle terminalen som er i forgrunnen

deallocvt Fjerner ubrukte virtuelle terminaler

dumpkeys Dumper tastaturoversettelsestabellene

fgconsole Skriver ut nummeret til den aktive virtuelle terminalen

getkeycodes Skriver ut kjernens skanningskode til nøkkelkode tilordningstabellen

kbdinfo Får informasjon om statusen til en konsoll

kbd_mode Rapporterer eller stiller inn tastaturmodus

kbdrate Stiller inn repetisjons- og forsinkelseshastigheter for tastaturet

loadkeys Laster tastaturoversettelsestabellene

loadunimap Laster kjernens unicode til font kartleggingstabellen

mapscrn Et utdatert program som pleide å laste en brukerdefinert utgang tegnkartleggingstabell i konsolldriveren; dette er nå gjort av **setfont**

openvt Starter et program på en ny virtuell terminal (VT)

psfaddtable Legger til en Unicode tegntabell til en konsollfont

psfgettable Trekker ut den innebygde Unicode tegntabellen fra en konsoll font

psfstriptable Fjerner den innebygde Unicode tegntabellen fra en konsoll font

psfxtable Håndterer Unicode tegntabeller for konsollfonter

setfont Endrer den forbedrede grafikkadapteren (EGA) og videografikk matrise (VGA) fonter på konsollen

setkeycodes Laster kjernens skanningskode til nøkkelkode kartleggingstabelloppføringer; dette er nyttig hvis det er uvanlige taster på tastaturet

setleds Stiller inn tastaturflagg og lysdioder (LED)

setmetamode Definerer tastaturets metanøkkel håndtering

setvtrgb Stiller inn konsollfargekartet i alle virtuelle terminaler

showconsolefont Viser gjeldende EGA/VGA konsollskjermskrift

showkey Rapporterer skanningskodene, nøkkelkodene og ASCII-kodene til tastene som trykkes på tastaturet

unicode_start Setter tastaturet og konsollen i UNICODE modus [Ikke bruk dette programmet med mindre tastaturfilen er i ISO-8859-1-kodingen. Til andre kodinger, gir dette verktøyet feil resultater.]

unicode_stop Tilbakestiller tastatur og konsoll fra UNICODE modus

8.63. Libpipeline-1.5.5

Libpipeline pakken inneholder et bibliotek for å manipulere kanaler av delprosesser på en fleksibel og praktisk måte.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 9.7 MB

8.63.1. Installasjon av Libpipeline

Forbered Libpipeline for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.63.2. Innhold i Libpipeline

Installert bibliotek: libpipeline.so

Korte beskrivelser

`libpipeline` Dette biblioteket brukes til å trygt konstruere kanaler mellom delprosesser

8.64. Make-4.3

Make pakken inneholder et program for å kontrollere genereringen av kjørbare filer og andre ikke-kildefiler av en pakke fra kildefiler.

Omtrentlig byggetid: 0.5 SBU
Nødvendig diskplass: 13 MB

8.64.1. Installasjon av Make

Forbered Make for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.64.2. Innhold i Make

Installert program: make

Korte beskrivelser

make Avgjør automatisk hvilke deler av en pakke som må bli (re)kompilert og utsteder deretter de relevante kommandoene

8.65. Patch-2.7.6

Patch pakken inneholder et program for å endre eller lage filer ved å bruke en “patch” fil som vanligvis opprettes av **diff** programmet.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 12 MB

8.65.1. Installasjon av patch

Forbered patch for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.65.2. Innhold i oppdateringen

Installert program: patch

Korte beskrivelser

patch Endrer filer i henhold til en patch fil (En patch fil er normalt en forskjellsoppføring opprettet med **diff** programmet. Ved å bruke disse forskjellene på originalfilene, **patch** oppretter de lappede versjonene.)

8.66. Tar-1.34

Tar pakken gir muligheten til å lage tar arkiver også å utføre forskjellige andre typer arkivmanipulering. Tar kan brukes på tidligere opprettede arkiver for å trekke ut filer, for å lagre flere filer, eller for å oppdatere eller liste filer som allerede var lagret.

Omtrentlig byggetid: 1.7 SBU
Nødvendig diskplass: 40 MB

8.66.1. Installasjon av Tar

Forbered Tar for kompilering:

```
FORCE_UNSAFE_CONFIGURE=1  \
./configure --prefix=/usr
```

Betydningen av konfigureringsalternativet:

`FORCE_UNSAFE_CONFIGURE=1`
Dette tvinger testen for `mknod` å bli kjørt som `root`. Det anses generelt som farlig å kjøre denne testen som `root` bruker, men siden den kjøres på et system som kun er delvis bygget, å overstyre det er OK.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

En test, capabilities: binary store/restore, er kjent for å mislykkes hvis den kjøres (LFS mangler selinux), men vil bli hoppet over hvis vertskjernen ikke støtter utvidede attributter på filsystemet som brukes til å bygge LFS.

Installer pakken:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.34
```

8.66.2. Innhold i Tar

Installerte programmer: tar
Installert mappe: /usr/share/doc/tar-1.34

Korte beskrivelser

tar Oppretter, trekker ut filer fra og viser innholdet i arkiver, også kjent som tarballs

8.67. Texinfo-6.8

Texinfo pakken inneholder programmer for lesing, skriving og konvertere informasjonssider.

Omtrentlig byggetid: 0.6 SBU
Nødvendig diskplass: 112 MB

8.67.1. Installasjon av Texinfo

Forbered Texinfo for kompilering:

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

Igjen, fiks et problem med å bygge pakken med Glibc-2.34 eller nyere:

```
sed -e 's/__attribute_nonnull__/__nonnull/' \
    -i gnulib/lib/malloc/dynarray-skeleton.c
```

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

Installer eventuelt komponentene som hører til i en TeX installasjon:

```
make TEXMF=/usr/share/texmf install-tex
```

Betydningen av make parameteren:

TEXMF=/usr/share/texmf

TEXMF makefile variabelen holder plasseringen av roten til TeX treet hvis for eksempel en TeX pakke vil bli installert senere.

Infodokumentasjonssystemet bruker en ren tekstfil til å holde listen over menyoppføringer. Filen ligger på `/usr/share/info/dir`. Dessverre, på grunn av sporadiske problemer i Makefiles for forskjellige pakker, kan det noen ganger gå ut av synkronisering med infosidene som er installert på systemet. Hvis `/usr/share/info/dir` filen noen gang trenger å bli gjenskapt, vil følgende valgfrie kommandoer utføre oppgaven:

```
pushd /usr/share/info
  rm -v dir
  for f in *
    do install-info $f dir 2>/dev/null
  done
popd
```

8.67.2. Innhold i Texinfo

Installerte programmer: info, install-info, makeinfo (link to texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf, og texindex
Installert bibliotek: MiscXS.so, Parsetexi.so, og XSParagraph.so (all in /usr/lib/texinfo)
Installerte mapper: /usr/share/texinfo og /usr/lib/texinfo

Korte beskrivelser

info Brukes til å lese informasjonssider som ligner på man sider, men går ofte mye dypere enn bare å forklare alle tilgjengelige kommandoers linjealternativer [For eksempel, sammenlign **man bison** og **info bison**.]

install-info Brukes til å installere infosider; den oppdaterer oppføringer i **info** indeksfil

makeinfo Oversetter de gitte Texinfo kildedokumentene til infosider, ren tekst eller HTML

pdftexi2dvi Brukes til å formatere det gitte Texinfo dokumentet til en Flyttbart dokumentformat (PDF) fil

pod2texi Konverterer Pod til Texinfo format

texi2any Oversett Texinfo kildedokumentasjon til forskjellige andre formater

texi2dvi Brukes til å formatere det gitte Texinfo dokumentet til en enhetsuavhengig fil som kan skrives ut

texi2pdf Brukes til å formatere det gitte Texinfo dokumentet til en Flyttbart dokumentformat (PDF) fil

texindex Brukes til å sortere Texinfo indeksfiler

8.68. Vim-8.2.4383

Vim pakken inneholder en kraftig tekstredigerer.

Omtrentlig byggetid: 2.4 SBU
Nødvendig diskplass: 206 MB

Alternativer til Vim

Hvis du foretrekker en annen tekstredigerer—som Emacs, Joe, eller Nano—Vennligst se *https://www.linuxfromscratch.org/blfs/view/stable-systemd/postlfs/editors.html* for foreslåtte installasjonsinstruksjoner.

8.68.1. Installasjon av Vim

Først endrer du standardplasseringen for `vimrc` konfigurasjonsfil til `/etc`:

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Forbered vim for kompilering:

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

Kompiler pakken:

```
make
```

For å forberede testene, sørg for at brukeren `tester` kan skrive til kildetreet:

```
chown -Rv tester .
```

Kjør nå testene som bruker `tester`:

```
su tester -c "LANG=en_US.UTF-8 make -j1 test" &> vim-test.log
```

Testpakken sender ut mange binære data til skjermen. Dette kan forårsake problemer med innstillingene til gjeldende terminal. Problemet kan unngås ved å omdirigere utdataene til en loggfil som vist ovenfor. En vellykket test vil resultere i ordene "ALL DONE" i loggfilen ved ferdigstillelse.

Installer pakken:

```
make install
```

Mange brukere er vant til å bruke **vi** i stedet for **vim**. For å tillate kjøringen av **vim** når brukere vanligvis skriver **vi**, lage en symbolkobling for både binærsiden og man siden i det angitte språket:

```
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
```

Som standard er vims dokumentasjon installert i `/usr/share/vim`. Følgende symbolkobling gjør det mulig å få tilgang til dokumentasjonen via `/usr/share/doc/vim-8.2.4383`, gjør det samsvar med plasseringen av dokumentasjonen for andre pakker:

```
ln -sv ../vim/vim82/doc /usr/share/doc/vim-8.2.4383
```

Hvis et X Window System skal installeres på LFS systemet, kan det være nødvendig å rekompilere vim etter installasjon av X. Vim kommer med en GUI versjon av tekstredigereren som krever X og noen flere biblioteker som skal installeres. For mer informasjon om denne prosessen, se vim dokumentasjonen og vim installasjonssiden i BLFS boka på *https://www.linuxfromscratch.org/blfs/view/stable-systemd/postlfs/vim.html*.

8.68.2. Konfigurerer Vim

Som standard, **vim** kjører i vi inkompatibel modus. Dette kan være nytt for brukere som har brukt andre tekstredigerere tidligere. “nocompatible” innstillingen er inkludert nedenfor for å fremheve faktum at en ny atferd blir brukt. Det minner også de som ville endre til “compatible” modus at det skal være den første innstilling i konfigurasjonsfilen. Dette er nødvendig fordi det endrer andre innstillinger og overstyringer må komme etter denne innstillingen. Opprett en standard **vim** konfigurasjonsfil ved å kjøre følgende:

```
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
```

`set nocompatible` innstillingen gjør at **vim** oppfører seg på en mer nyttig måte (standard) enn vi kompatibel måte. Fjern “no” å beholde det gamle **vi** oppførselen. *`set backspace=2`* innstillingen tillater tilbaketast over linjeskift, autoinnrykk og starten på et innlegg. *`syntax on`* parameter aktiverer vim sin syntaks fremheving. *`set mouse=`* innstillingen aktiverer riktig liming av tekst med musen når du jobber i chroot eller over en ekstern tilkobling. Endelig, *if* erklæring med *`set background=dark`* innstillingen korrigerer **vim**'s gjetting om bakgrunnsfargen til en eller annen terminal emulatorer. Dette gir uthevingen et bedre fargevalg for bruk på svart bakgrunn for disse programmene.

Dokumentasjon for andre tilgjengelige alternativer kan fås ved å kjører følgende kommando:

```
vim -c ':options'
```

Note

Som standard installerer vim kun stavefiler for det engelske språket. For å installere stavefiler for ditt foretrukne språk, last ned `*.spl` og eventuelt `*.sug` filer for ditt språk og tegnkoding fra *ftp://ftp.vim.org/pub/vim/runtime/spell/* og lagre dem til `/usr/share/vim/vim82/spell/`.

For å bruke disse stavefilene, noen konfigurasjoner i `/etc/vimrc` trengs, f.eks.:

```
set spelllang=en,ru
set spell
```

For mer informasjon, se den aktuelle README filen på URLen ovenfor.

8.68.3. Innhold i Vim

Installerte programmer: ex (lenker til vim), rview (lenker til vim), rvim (lenker til vim), vi (lenker til vim), view (lenker til vim), vim, vimdiff (lenker til vim), vimtutor, og xxd
Installert mappe: /usr/share/vim

Korte beskrivelser

ex Starter **vim** i ex modus

rview Er en begrenset versjon av **view**; ikke noen skall kommandoer kan startes og **view** kan ikke suspenderes

rvim Er en begrenset versjon av **vim**; ikke noen skall kommandoer kan startes og **vim** kan ikke suspenderes

vi Lenker til **vim**

view Starter **vim** i skrivebeskyttet modus

vim Er tekstredigereren

vimdiff Redigerer to eller tre versjoner av en fil med **vim** og viser forskjellene

vimtutor Lærer de grunnleggende tastene og kommandoene til **vim**

xxd Oppretter en hex dump av den gitte filen; den kan også gjøre det motsatte, slik at det kan brukes til binær endring

8.69. MarkupSafe-2.0.1

MarkupSafe er en Python modul som implementerer en XML/HTML/XHTML Markup sikker streng.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 520 KB

8.69.1. Installasjon av MarkupSafe

Kompiler MarkupSafe med følgende kommando:

```
python3 setup.py build
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
python3 setup.py install --optimize=1
```

8.69.2. Innhold i MarkupSafe

Installert katalog: /usr/lib/python3.10/site-packages/MarkupSafe-2.0.1-py3.10.egg

8.70. Jinja2-3.0.3

Jinja2 er en Pythonmodul som implementerer et enkelt pytonisk malspråk.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 3.7 MB

8.70.1. Installasjon av Jinja2

Installer pakken:

```
python3 setup.py install --optimize=1
```

8.70.2. Innhold i Jinja2

Installerte mapper: /usr/lib/python3.10/site-packages/Jinja2-3.0.3-py3.10.egg

8.71. Systemd-250

Systemd pakken inneholder programmer for å kontrollere oppstarten, kjøring og avslutning av systemet.

Omtrentlig byggetid: 2.2 SBU
Nødvendig diskplass: 273 MB

8.71.1. Installasjon av systemd

Bruk først en oppdatering for å fikse en sikkerhetssårbarhet og regresjoner med vertsnavn og inaktive enheter:

```
patch -Np1 -i ../systemd-250-upstream_fixes-1.patch
```

Fjern to unødvendige grupper, `render` og `sgx`, fra standard udev regler:

```
sed -i -e 's/GROUP="render"/GROUP="video"/' \
       -e 's/GROUP="sgx", //' rules.d/50-udev-default.rules.in
```

Forbered systemd for kompilering:

```
mkdir -p build
cd       build

meson --prefix=/usr                 \
      --sysconfdir=/etc             \
      --localstatedir=/var          \
      --buildtype=release           \
      -Dblkid=true                  \
      -Ddefault-dnssec=no           \
      -Dfirstboot=false             \
      -Dinstall-tests=false         \
      -Dldconfig=false              \
      -Dsysusers=false              \
      -Db_lto=false                 \
      -Drpmmacrosdir=no             \
      -Dhomed=false                 \
      -Duserdb=false                \
      -Dman=false                   \
      -Dmode=release                \
      -Ddocdir=/usr/share/doc/systemd-250 \
      ..
```

Betydningen av meson alternativene:

`--buildtype=release`
Denne bryteren overstyrer standard byggetype (“debug”), som ville produsert uoptimaliserte binære filer.

`-Ddefault-dnssec=no`
Denne bryteren slår av den eksperimentelle DNSSEC støtten.

`-Dfirstboot=false`
Denne bryteren forhindrer installasjon av systemd tjenester ansvarlig for å sette opp systemet for den første gangen. De er ikke nyttige for LFS pga alt gjøres manuelt.

-Dinstall-tests=false

Denne bryteren forhindrer installasjon av de kompilerte testene.

-Dldconfig=false

Denne bryteren forhindrer installasjon av en systemd enhet som kjører **ldconfig** ved oppstart, som ikke er nyttig for kildedistribusjoner som LFS og gjør oppstartstiden lengre. Fjern det hvis den beskrevne funksjonen er ønsket.

-Dsysusers=false

Denne bryteren forhindrer installasjon av systemd tjenester som er ansvarlige for å sette opp `/etc/group` og `/etc/passwd` filer. Begge filene ble opprettet i forrige kapittel. Denne nissen (daemon) er ikke nyttig på et LFS system siden brukerkontoer opprettes manuelt.

-Drpmmacrosdir=no

Denne bryteren deaktiverer installasjon av RPM makroer for bruk med systemd fordi LFS ikke støtter RPM.

-D{userdb,homed}=false

Fjern to nisser som har avhengigheter som ikke passer omfanget av LFS.

-Dman=false

Forhindre generering av man sider for å unngå ekstra avhengigheter. Vi vil installere forhåndsgenererte man sider for systemd fra en tarball senere.

-Dmode=release

Deaktiver noen funksjoner som anses som eksperimentelle av oppstrøms.

Kompiler pakken:

```
ninja
```

Installer pakken:

```
ninja install
```

Installer man sidene:

```
tar -xf ../../systemd-man-pages-250.tar.xz --strip-components=1 -C /usr/share/man
```

Fjern en ubrukelig mappe:

```
rm -rf /usr/lib/pam.d
```

Opprett `/etc/machine-id` filen som trengs av **systemd-journald**:

```
systemd-machine-id-setup
```

Sett opp den grunnleggende målstrukturen:

```
systemctl preset-all
```

8.71.2. Innhold i systemd

Installerte programmer: busctl, coredumpctl, halt (symlenke til systemctl), hostnamectl, init, journalctl, kernel-install, localectl, loginctl, machinectl, networkctl, oomctl, portablectl, poweroff (symlenke til systemctl), reboot (symlenke til systemctl), resolvconf (symlenke til resolvectl), resolvectl, runlevel (symlenke til systemctl), shutdown (symlenke til systemctl), systemctl, systemd-analyze, systemd-ask-password, systemd-cat, systemd-cgls, systemd-cgtop, systemd-creds, systemd-delta, systemd-detect-virt, systemd-dissect, systemd-escape, systemd-hwdb, systemd-id128, systemd-inhibit, systemd-machine-id-setup, systemd-mount, systemd-notify, systemd-nspawn, systemd-path, systemd-repart, systemd-resolve (symlenke til resolvectl), systemd-run, systemd-socket-activate, systemd-stdio-bridge, systemd-sysext, systemd-tmpfiles, systemd-tty-ask-password-agent, systemd-umount (symlenke til systemd-mount), telinit (symlenke til systemctl), timedatectl, og udevadm

Installerte biblioteker: libnss_myhostname.so.2, libnss_mymachines.so.2, libnss_resolve.so.2, libnss_systemd.so.2, libsystemd.so, libsystemd-shared-250.so (i /usr/lib/systemd), og libudev.so

Installerte mapper: /etc/binfmt.d, /etc/init.d, /etc/kernel, /etc/modules-load.d, /etc/sysctl.d, /etc/systemd, /etc/tmpfiles.d, /etc/udev, /etc/xdg/systemd, /usr/lib/systemd, /usr/lib/udev, /usr/include/systemd, /usr/lib/binfmt.d, /usr/lib/environment.d, /usr/lib/kernel, /usr/lib/modules-load.d, /usr/lib/sysctl.d, /usr/lib/systemd, /usr/lib/tmpfiles.d, /usr/share/doc/systemd-250, /usr/share/factory, /usr/share/systemd, /var/lib/systemd, og /var/log/journal

Korte beskrivelser

busctl Brukes til å selvransake og overvåke D-Bus bussen

coredumpctl Brukes til å hente kjernedumper fra systemd journalen

halt Starter vanligvis **shutdown** med `-h` alternativet, bortsett fra når du allerede er på kjørenivå 0, så ber den kjernen om å stoppe systemet; det noterer i filen `/var/log/wtmp` at systemet blir slått av

hostnamectl Brukes til å spørre og endre systemets vertsnavn og relaterte innstillinger

init Er den første prosessen som startes når kjernen har initialisert maskinvaren som tar over oppstartsprosessen og starter alle prosesser i henhold til konfigurasjonsfilene. I dette tilfellet starter den systemd

journalctl Brukes til å spørre om innholdet i systemd journalen

kernel-install Brukes til å legge til og fjerne kjerne- og initramfs-bilder til og fra /boot. I LFS gjøres dette manuelt

localectl Brukes til å spørre og endre systemlokaliteten og tastaturoppsettets innstillinger

loginctl Brukes til å selvransake og kontrollere tilstanden til systemd påloggings behandleren

machinectl Brukes til å selvransake og kontrollere tilstanden til systemd virtuelle maskin og container registreringsbehandler

networkctl Brukes til å selvransake og konfigurere nettverkets koblinger konfigurert av systemd networkd

oomctl	Styrer systemd tomt for minne (Out Of Memory) nissen
portablectl	Brukes til å koble til eller koble fra bærbare tjenester fra det lokale systemet
poweroff	Instruerer kjernen om å stoppe systemet og slå av datamaskinen (se **halt**)
reboot	Instruerer kjernen om å starte systemet på nytt (se **halt**)
resolvconf	Registrerer DNS server og domenekonfigurasjon med **systemd-resolved**
resolvectl	Sender kontrollkommandoer til nettverksnavnoppløsningens behandler, eller løser domenenavn, IPv4- og IPv6-adresser, DNS poster og tjenester
runlevel	Sender ut forrige og gjeldende kjøringsnivå, som nevnt i siste run-level oppføring i `/run/utmp`
shutdown	Får systemet ned på en trygg og sikker måte, signaliserer alle prosesser og varsle alle påloggede brukere
systemctl	Brukes til å selvransake og kontrollere tilstanden til systemd system og servicebehandler
systemd-analyze	Brukes til å bestemme systemoppstartsytelsen til gjeldende boot, samt identifisere plagsomme systemd enheter
systemd-ask-password	Brukes til å spørre et systempassord eller passordfrase fra brukeren, ved å bruke en spørsmålsmelding spesifisert på kommandolinjen
systemd-cat	Brukes til å koble til STDOUT og STDERR utdata til en prosess med systemd journal
systemd-cgls	Viser rekursivt innholdet i den valgte Linux kontrollens gruppehierarki i et tre
systemd-cgtop	Viser de øverste kontrollgruppene til den lokale Linux kontrollgruppens hierarki, sortert etter CPU, minne og disk I/O-belastning
systemd-creds	Viser og behandler akkreditiver
systemd-delta	Brukes til å identifisere og sammenligne konfigurasjonsfiler i `/etc` som overstyrer standard motparter i `/usr`
systemd-detect-virt	Oppdager om systemet kjøres i et virtuelt miljø, og justerer udev deretter
systemd-dissect	Brukes til å inspisere OS diskbilder
systemd-escape	Brukes til å verne strenger for inkludering i systemd enhets navn
systemd-hwdb	Brukes til å administrere maskinvaredatabasen (hwdb)
systemd-id128	Genererer og skriver ut id128 strenger
systemd-inhibit	Brukes til å kjøre et program med avstenging, hvilemodus eller inaktiv hemmerlås tatt, forhindrer en handling som for eksempel en systemavslutning til prosessen er fullført

systemd-machine-id-setup	Brukes av systeminstallasjonsverktøy for å initialisere maskin-ID lagret i `/etc/machine-id` ved installasjonstidspunktet med en tilfeldig generert ID
systemd-mount	Brukes til midlertidig montering eller automontering av disker
systemd-notify	Brukes av nisseskript for å varsle init-systemet om status endringer
systemd-nspawn	Brukes til å kjøre en kommando eller OS i et lett navneområde container
systemd-path	Brukes til å spørre system- og brukerstier
systemd-repart	Brukes til å vokse og legge til partisjoner til en partisjonstabell når systemd brukes i et OS bilde (f.eks. en container)
systemd-resolve	Brukes til å løse domenenavn, IPV4- og IPv6-adresser, DNS ressursposter og tjenester
systemd-run	Brukes til å opprette og starte en forbigående .service eller en .scope enhet og kjør den angitte kommandoen i den. Dette er nyttig for validering av systemenheter
systemd-socket-activate	Brukes til å lytte på socket enheter og starte en prosess med en vellykket tilkobling til en socket
systemd-sysext	Aktiverer systemutvidelsesbilder
systemd-tmpfiles	Oppretter, sletter og rydder opp i flyktige og midlertidige filer og mapper, basert på konfigurasjonsfilformatet og plasseringen spesifisert i `tmpfiles.d` mappene
systemd-umount	Demonterer monteringspunkter
systemd-tty-ask-password-agent	Brukes til å liste og/eller behandle ventende systemd passord forespørsler
telinit	Forteller **init** hvilket kjørenivå det skal endres til
timedatectl	Brukes til å spørre og endre systemklokken og dens innstillinger
udevadm	Er et generisk udev administrasjonsverktøy som kontrollerer udevd nissen, gir informasjon fra Udev maskinvaredatabasen, overvåker uevents, venter på at uevents skal fullføres, tester udev konfigurasjon og utløser uevents for en gitt enhet
`libsystemd`	Er det viktigste systemd verktøybiblioteket
`libudev`	Er et bibliotek for å få tilgang til Udev enhetsinformasjon

8.72. D-Bus-1.12.20

D-Bus er et meldingsbusssystem, en enkel måte for applikasjoner å snakke til hverandre. D-Bus leverer både en systemnisse (system daemon) (for hendelser som f.eks "ny maskinvareenhet lagt til" eller "skriverkø endret") og en per brukerpåloggingsøkt nisse (for generelle IPC behov blant brukerens applikasjoner). Dessuten er meldingsbussen bygget på toppen av et generelt en-til-en rammeverk for meldingsoverføring, som kan brukes av to applikasjoner til å kommunisere direkte (uten å gå gjennom meldingsbussnissen).

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 18 MB

8.72.1. Installasjon av D-Bus

Prepare D-Bus for compilation:

```
./configure --prefix=/usr                       \
            --sysconfdir=/etc                   \
            --localstatedir=/var                \
            --disable-static                    \
            --disable-doxygen-docs              \
            --disable-xml-docs                  \
            --docdir=/usr/share/doc/dbus-1.12.20 \
            --with-console-auth-dir=/run/console \
            --with-system-pid-file=/run/dbus/pid \
            --with-system-socket=/run/dbus/system_bus_socket
```

The meaning of the configure options:

`--with-console-auth-dir=/run/console`
Dette spesifiserer plasseringen av ConsoleKit autorisasjons mappen.

`--with-system-pid-file=/run/dbus/pid` og *`--with-system-socket=/run/dbus/system_bus_socket`*
Disse angir plasseringen av PID filen og systembusskontakten til å være i `/run`, i stedet for utdaterte `/var/run`.

Kompiler pakken:

```
make
```

Denne pakken kommer med en testpakke, men den krever flere pakker som ikke er inkludert i LFS. Instruksjoner for å kjøre testpakken finner du i BLFS-boken på *https://www.linuxfromscratch.org/blfs/view/stable-systemd/general/dbus.html*.

Installer pakken:

```
make install
```

Lag en symbolkobling slik at D-Bus og systemd kan bruke den samme `machine-id` filen:

```
ln -sfv /etc/machine-id /var/lib/dbus
```

8.72.2. Innhold i D-Bus

Installerte programmer: dbus-cleanup-sockets, dbus-daemon, dbus-launch, dbus-monitor, dbus-run-session, dbus-send, dbus-test-tool, dbus-update-activation-environment, og dbus-uuidgen
Installerte biblioteker: libdbus-1.{a,so}
Installerte mapper: /etc/dbus-1, /usr/include/dbus-1.0, /usr/lib/dbus-1.0, /usr/share/dbus-1, /usr/share/doc/dbus-1.12.20, og /var/lib/dbus

Korte beskrivelser

dbus-cleanup-sockets	brukes til å fjerne gjenværende socket i en mappe
dbus-daemon	Er D-Bus-meldingsbussnisse
dbus-launch	Starter **dbus-daemon** fra et skall skript
dbus-monitor	Overvåker meldinger som går gjennom en D-Bus meldingsbuss
dbus-run-session	Starter en øktbussforekomst av **dbus-daemon** fra et skallskript og starter et spesifisert program i den økten
dbus-send	Sender en melding til en D-Bus meldingsbuss
dbus-test-tool	Er et verktøy for å hjelpe pakker å teste D-Bus
dbus-update-activation-environment	Oppdaterer miljøvariabler som vil bli satt for D-Bus økttjenester
dbus-uuidgen	Genererer en universell unik ID
`libdbus-1`	Inneholder API funksjoner som brukes til å kommunisere med meldingsbussen til D-bus

8.73. Man-DB-2.10.1

Man-DB pakken inneholder programmer for å finne og se på man sider.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 39 MB

8.73.1. Installasjon av Man-DB

Forbered Man-DB for kompilering:

```
./configure --prefix=/usr                         \
            --docdir=/usr/share/doc/man-db-2.10.1 \
            --sysconfdir=/etc                     \
            --disable-setuid                      \
            --enable-cache-owner=bin              \
            --with-browser=/usr/bin/lynx          \
            --with-vgrind=/usr/bin/vgrind         \
            --with-grap=/usr/bin/grap
```

Betydningen av konfigureringsalternativene:

`--disable-setuid`
Dette deaktiverer å lage **man** program setuid til bruker `man`.

`--enable-cache-owner=bin`
Dette gjør at de systemomfattende hurtigbufferfilene eies av brukeren `bin`.

`--with-...`
Disse tre parameterne brukes til å angi noen standardprogrammer. **lynx** er en tekstbasert nettleser (se BLFS for installasjonsinstruksjoner), **vgrind** konverterer programkilder til Groff inndata, og **grap** er nyttig for å sette grafer i Groff dokumenter. **vgrind** og **grap** programmer er vanligvis ikke nødvendig for å vise man sider. De er ikke en del av LFS eller BLFS, men du bør kunne installere dem selv etter at du har fullført LFS hvis du ønsker å gjøre det.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.73.2. Ikke-engelske manualsider i LFS

Følgende tabell viser tegnsettet som Man-DB antar manuelle sider installert under `/usr/share/man/<ll>` vil være kodet med. I tillegg til dette, bestemmer Man-DB korrekt om manualsider installert i den katalogen er UTF-8-kodet.

Table 8.1. Forventet tegnkoding av eldre 8-biters manualsider

Language (code)	Encoding	Language (code)	Encoding
Danish (da)	ISO-8859-1	Croatian (hr)	ISO-8859-2

Language (code)	Encoding	Language (code)	Encoding
German (de)	ISO-8859-1	Hungarian (hu)	ISO-8859-2
English (en)	ISO-8859-1	Japanese (ja)	EUC-JP
Spanish (es)	ISO-8859-1	Korean (ko)	EUC-KR
Estonian (et)	ISO-8859-1	Lithuanian (lt)	ISO-8859-13
Finnish (fi)	ISO-8859-1	Latvian (lv)	ISO-8859-13
French (fr)	ISO-8859-1	Macedonian (mk)	ISO-8859-5
Irish (ga)	ISO-8859-1	Polish (pl)	ISO-8859-2
Galician (gl)	ISO-8859-1	Romanian (ro)	ISO-8859-2
Indonesian (id)	ISO-8859-1	Russian (ru)	KOI8-R
Icelandic (is)	ISO-8859-1	Slovak (sk)	ISO-8859-2
Italian (it)	ISO-8859-1	Slovenian (sl)	ISO-8859-2
Norwegian Bokmal (nb)	ISO-8859-1	Serbian Latin (sr@latin)	ISO-8859-2
Dutch (nl)	ISO-8859-1	Serbian (sr)	ISO-8859-5
Norwegian Nynorsk (nn)	ISO-8859-1	Turkish (tr)	ISO-8859-9
Norwegian (no)	ISO-8859-1	Ukrainian (uk)	KOI8-U
Portuguese (pt)	ISO-8859-1	Vietnamese (vi)	TCVN5712-1
Swedish (sv)	ISO-8859-1	Simplified Chinese (zh_CN)	GBK
Belarusian (be)	CP1251	Simplified Chinese, Singapore (zh_SG)	GBK
Bulgarian (bg)	CP1251	Traditional Chinese, Hong Kong (zh_HK)	BIG5HKSCS
Czech (cs)	ISO-8859-2	Traditional Chinese (zh_TW)	BIG5
Greek (el)	ISO-8859-7		

Note

Manualsider på språk som ikke er på listen støttes ikke.

8.73.3. Innhold i Man-DB

Installerte programmer: accessdb, apropos (lenker til whatis), catman, lexgrog, man, man-recode, mandb, manpath, og whatis
Installerte biblioteker: libman.so og libmandb.so (begge i /usr/lib/man-db)
Installerte mapper: /usr/lib/man-db, /usr/libexec/man-db, og /usr/share/doc/man-db-2.10.1

Korte beskrivelser

accessdb Dumper **whatis** databaseinnhold i menneskelig lesbar form

apropos	Søker **whatis** database og viser de korte beskrivelsene av systemkommandoer som inneholder en gitt streng
catman	Oppretter eller oppdaterer de forhåndsformaterte manualsidene
lexgrog	Viser én-linjes sammendragsinformasjon om en gitt manualside
man	Formaterer og viser den forespurte manualsiden
man-recode	Konverterer manualsider til en annen koding
mandb	Oppretter eller oppdaterer **whatis** databasen
manpath	Viser innholdet i $MANPATH eller (hvis $MANPATH ikke er angitt) en passende søkebane basert på innstillingene i man.conf og brukerens miljø
whatis	Søker **whatis** databasen og viser de korte beskrivelsene av systemkommandoer som inneholder de gitte nøkkelord som et eget ord
`libman`	Inneholder kjøretidsstøtte for **man**
`libmandb`	Inneholder kjøretidsstøtte for **man**

8.74. Procps-ng-3.3.17

Procps-ng pakken inneholder programmer for overvåking av prosesser.

Note

Denne pakken trekkes ut til mappen `procps-3.3.17`, ikke det forventede `procps-ng-3.3.17`.

Omtrentlig byggetid: 0.4 SBU
Nødvendig diskplass: 19 MB

8.74.1. Installasjon av Procps-ng

Forbered procps-ng for kompilering:

```
./configure --prefix=/usr                            \
            --docdir=/usr/share/doc/procps-ng-3.3.17 \
            --disable-static                         \
            --disable-kill                           \
            --with-systemd
```

Betydningen av konfigureringsalternativet:

`--disable-kill`
Denne bryteren deaktiverer bygging av **kill** kommandoen som vil bli installert av Util-linux pakken.

Kompiler pakken:

```
make
```

For å kjøre testpakken, kjør:

```
make check
```

Fem tester relatert til pkill er kjent for å mislykkes på grunn av et problem med tester som ikke ble oppdatert.

Installer pakken:

```
make install
```

8.74.2. Innhold i Procps-ng

Installerte programmer: free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w, og watch
Installert bibliotek: libprocps.so
Installerte mapper: /usr/include/proc og /usr/share/doc/procps-ng-3.3.17

Korte beskrivelser

free Rapporterer mengden ledig og brukt minne (både fysisk og vekselminne) i systemet

pgrep Slår opp prosesser basert på deres navn og andre attributter

pidof Rapporterer PID-ene til de gitte programmene

pkill Signaliserer prosesser basert på deres navn og andre attributter

pmap Rapporterer minnekartet for den gitte prosessen

ps Viser gjeldende prosesser

pwait Venter til en prosess er ferdig før den utføres.

pwdx Rapporterer gjeldende arbeidsmappe for en prosess

slabtop Viser detaljert kjerne platebuffer (slab cache) informasjon i sanntid

sysctl Endrer kjerneparametere under kjøretid

tload Skriver ut en graf over gjeldende systembelastningsgjennomsnitt

top Viser en liste over de mest CPU intensive prosessene; den gir en kontinuerlig titt på prosessoraktivitet i sanntid

uptime Rapporterer hvor lenge systemet har kjørt, hvor mange brukere det er logget på, og systemets belastningsgjennomsnitt

vmstat Rapporterer virtuelt minnestatistikk, gir informasjon om prosesser, minne, søking, blokk Input/Output (IO), feller og CPU aktivitet

w Viser hvilke brukere som for øyeblikket er pålogget, hvor og siden når

watch Kjører en gitt kommando gjentatte ganger, og viser den første skjermen full av utdata; dette lar en bruker se utdataens endring over tid

`libprocps` Inneholder funksjonene som brukes av de fleste programmer i denne pakken

8.75. Util-linux-2.37.4

Util-linux pakken inneholder diverse hjelpeprogrammer. Blant dem er verktøy for håndtering av filsystemer, konsoller, partisjoner, og meldinger.

Omtrentlig byggetid: 1.1 SBU
Nødvendig diskplass: 261 MB

8.75.1. Installasjon av Util-linux

Forbered Util-linux for kompilering:

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime   \
            --bindir=/usr/bin    \
            --libdir=/usr/lib    \
            --sbindir=/usr/sbin  \
            --docdir=/usr/share/doc/util-linux-2.37.4 \
            --disable-chfn-chsh  \
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
            --disable-runuser    \
            --disable-pylibmount \
            --disable-static     \
            --without-python
```

Alternativene --disable og --without forhindrer advarsler om bygningskomponenter som krever pakker som ikke er i LFS eller er inkonsistent med programmer installert av andre pakker.

Kompiler pakken:

```
make
```

Hvis ønskelig, kjør testpakken som en ikke-root bruker:

Warning

Å kjører testpakken som `root` bruker kan være skadelig for systemet ditt. For å kjøre den, må CONFIG_SCSI_DEBUG alternativet for kjernen være tilgjengelig i det gjeldende systemet og må bygges som en modul. Å bygge den inn i kjernen vil forhindre oppstart. For komplett dekning, må andre BLFS pakker installeres. Om ønskelig kan denne testen kjøres etter omstart i det fullførte LFS systemet og med å kjøre:

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

Note

Det er en test som mislykkes i chroot miljøet og får testene til å henge for alltid. Problemet oppstår ikke utenfor chroot miljøet. For å omgå problemet, slett testen:

```
rm tests/ts/lsns/ioctl_ns
```

```
chown -Rv tester .
su tester -c "make -k check"
```

Installer pakken:

```
make install
```

8.75.2. Innhold i Util-linux

Installerte programmer: addpart, agetty, blkdiscard, blkid, blkzone, blockdev, cal, cfdisk, chcpu, chmem, choom, chrt, col, colcrt, colrm, column, ctrlaltdel, delpart, dmesg, eject, fallocate, fdisk, fincore, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipcs, irqtop, isosize, kill, last, lastb (lenker til last), ldattach, linux32, linux64, logger, look, losetup, lsblk, lscpu, lsipc, lsirq, 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, sulogin, swaplabel, swapoff (lenker til swapon), swapon, switch_root, taskset, uclampset, ul, umount, uname26, unshare, utmpdump, uuidd, uuidgen, uuidparse, wall, wdctl, whereis, wipefs, x86_64, og zramctl

Installerte biblioteker: libblkid.so, libfdisk.so, libmount.so, libsmartcols.so, og libuuid.so

Installerte mapper: /usr/include/blkid, /usr/include/libfdisk, /usr/include/libmount, /usr/include/libsmartcols, /usr/include/uuid, /usr/share/doc/util-linux-2.37.4, og /var/lib/hwclock

Korte beskrivelser

addpart Informerer Linux kjernen om nye partisjoner

agetty Åpner en tty port, ber om et påloggingsnavn, og starter deretter **login** programmet

blkdiscard Forkaster sektorer på en enhet

blkid Et kommandolinjeverktøy for å finne og skrive ut blokkenhets attributter

blkzone Kjører zone kommandoen på den gitte blokkenheten

blockdev Lar brukere kalle blokkenhet ioctls fra kommando linjen

cal Viser en enkel kalender

cfdisk Manipulerer partisjonstabellen til den gitte enheten

chcpu Endrer tilstanden til CPUer

chmem Konfigurerer minne

choom Viser og justerer OOM-killer poeng

chrt Manipulerer sanntidsattributter til en prosess

col Filtrerer ut omvendt linjemating

colcrt Filtrerer **nroff** utdata for terminaler som mangler noen muligheter, for eksempel overslag og halvlinjer

colrm Filtrerer ut de gitte kolonnene

column Formaterer en gitt fil i flere kolonner

ctrlaltdel Setter funksjonen til tastekombinasjonen Ctrl+Alt+Del til en hard eller myk tilbakestilling

delpart Ber Linux kjernen om å fjerne en partisjon

dmesg Dumper kjerneoppstartsmeldingene

eject Løser ut flyttbare medier

fallocate Forhåndstildeler plass til en fil

fdisk Manipulerer partisjonstabellen til den gitte enheten

fincore Teller sider med filinnhold i kjernen

findfs Finner et filsystem etter etikett eller universell unik identifikator (UUID)

findmnt Er et kommandolinjegrensesnitt til libmount biblioteket for arbeid med mountinfo, fstab og mtab filer

flock Skaffer en fillås og utfører deretter en kommando med låsen holdt

fsck Brukes til å sjekke, og eventuelt reparere, filsystemer

fsck.cramfs Performs a consistency check on the Cramfs file system on the given device

fsck.minix Utfører en konsistenssjekk på Minix filsystem på gitt enhet

fsfreeze Er en veldig enkel innpakning rundt FIFREEZE/FITHAW ioctl kjernedriveroperasjoner

fstrim Forkaster ubrukte blokker på et montert filsystem

getopt Analyserer alternativer i den gitte kommandolinjen

hexdump Dumper den gitte filen i heksadesimal eller i et annen gitt format

hwclock Leser eller stiller inn systemets maskinvareklokke, også kalt sanntidsklokken (RTC) eller grunnleggende inndata-utdata system (BIOS) klokke

i386 En symbolsk lenke til setarch

ionice Henter eller setter io planleggingsklasse og prioritet for et program

ipcmk Oppretter forskjellige IPC ressurser

ipcrm Fjerner den gitte IPC ressursen (Inter-Process Communication)

ipcs Gir IPC statusinformasjon

irqtop Viser informasjon om kjerneavbruddsteller i `top(1)` stilvisning

isosize Rapporterer størrelsen på et ISO9660-filsystem

kill Sender signaler til prosesser

last Viser hvilke brukere som sist logget på (og ut), søker tilbake gjennom `/var/log/wtmp` filen; den viser også systemoppstart, systemavslutning og endringer på kjørenivå

lastb Viser mislykkede påloggingsforsøk, som logget i `/var/log/btmp`

ldattach Fester en linjedisiplin til en seriellinje

linux32 En symbolsk lenke til setarch

linux64 En symbolsk lenke til setarch

logger Legger inn den gitte meldingen i systemloggen

look Viser linjer som begynner med den gitte strengen

losetup Setter opp og kontrollerer sløyfeenheter

lsblk Viser informasjon om alle eller utvalgte blokkenheter i et trelignende format

lscpu Skriver ut CPU arkitekturinformasjon

lsipc Skriver ut informasjon om IPC fasiliteter som for øyeblikket brukes i systemet

lsirq Viser informasjon om kjerneavbruddsteller

lslocks Viser lokale systemlåser

lslogins Viser informasjon om brukere, grupper og systemkontoer

lsmem Viser utvalg av tilgjengelig minne med deres tilkoblede status

lsns Viser navnerom

mcookie Genererer magiske informasjonskapsler (128-bit tilfeldige heksadesimale tall) for **xauth**

mesg Styrer om andre brukere kan sende meldinger til den gjeldende brukers terminal

mkfs Bygger et filsystem på en enhet (vanligvis en harddisk partisjon)

mkfs.bfs Oppretter et Santa Cruz Operations (SCO) bfs filsystem

mkfs.cramfs Oppretter et cramfs filsystem

mkfs.minix Oppretter et Minix filsystem

mkswap Initialiserer den gitte enheten eller filen til å brukes som et vekselminne område

more Et filter for å bla gjennom tekst et skjermbilde om gangen

mount Fester filsystemet på den gitte enheten til en spesifisert mappe i filsystemtreet

mountpoint Sjekker om mappen er et monteringspunkt

namei Viser de symbolske koblingene i de gitte stinavnene

nsenter Kjører et program med navnerom for andre prosesser

partx Forteller kjernen om tilstedeværelsen og nummereringen av diskens partisjoner

pivot_root Gjør det gitte filsystemet til det nye rotfilsystemet i gjeldende prosess

prlimit Få og angi ressursgrenser til en prosess

readprofile Leser informasjon om kjerneprofileringen

rename Gi nytt navn til de gitte filene, erstatter en gitt streng med en annen

renice Endrer prioriteten til kjørende prosesser

resizepart Ber Linux kjernen om å endre størrelsen på en partisjon

rev Reverserer linjene til en gitt fil

rkfill Verktøy for å aktivere og deaktivere trådløse enheter

rtcwake Brukes til å gå inn i systemets hviletilstand til et spesifisert vekkingstidspunkt

script Lager et typeskript av en terminaløkt

scriptlive	Kjøret sesjonens typeskript på nytt ved å bruke tidsinformasjon
scriptreplay	Spiller av typeskript ved hjelp av tidsinformasjon
setarch	Endringer rapportert arkitektur i et nytt programmiljø og setter personlighetsflagg
setsid	Kjører det gitte programmet i en ny økt
setterm	Angir terminalattributter
sfdisk	En manipulator for diskpartisjonstabeller
sulogin	Tillater `root` å logge inn; den er normalt startet av **init** når systemet går i enkeltbrukermodus
swaplabel	Gjør det mulig å endre vekselmonneområdets UUID og etikett
swapoff	Deaktiverer enheter og filer for søking og bruk av vekselminne
swapon	Aktiverer enheter og filer for søking og bruk av vekselminne og viser enhetene og filene som er i bruk
switch_root	Bytter til et annet filsystem som roten til monteringstreet
taskset	Henter eller setter en prosess sin CPU tilhørighet
uclampset	Manipuler bruk av clamping attributtene til systemet eller en prosess
ul	Et filter for å oversette understrek til skiftesekvenser som indikerer understreking for terminalen som er i bruk
umount	Kobler et filsystem fra systemets filtre
uname26	En symbolsk lenke til setarch
unshare	Kjører et program med noen navnerom som ikke er delt fra overordnet
utmpdump	Viser innholdet i den gitte påloggingsfilen i et mer brukervennlig format
uuidd	En nisse som brukes av UUID biblioteket for å generere tidsbasert UUID på en sikker og garantert unik måte
uuidgen	Oppretter nye UUID. Hver ny UUID kan med rimelighet vurderes unik blant alle UUID som er opprettet, på det lokale systemet og på andre systemer, i fortiden og i fremtiden
uuidparse	Et verktøy for å analysere unike identifikatorer
wall	Viser innholdet i en fil eller, som standard, dens standard inndata, på terminalene til alle påloggede brukere
wdctl	Viser maskinvareovervåkingsstatus
whereis	Rapporterer plasseringen av binær, kilde og man siden for den gitte kommandoen
wipefs	Sletter en filsystemsignatur fra en enhet
x86_64	En symbolsk lenke til setarch
zramctl	Et program for å sette opp og kontrollere zram (komprimert ram disk) enheter
`libblkid`	Inneholder rutiner for enhetsidentifikasjon og symbol utdrag
`libfdisk`	Inneholder rutiner for manipulering av partisjonstabeller
`libmount`	Inneholder rutiner for montering og avmontering av en blokkenhet
`libsmartcols`	Inneholder rutiner for å hjelpe skjermutdata i tabulatorform
`libuuid`	Inneholder rutiner for å generere unike identifikatorer for objekter som kan være tilgjengelig utenfor det lokale systemet

8.76. E2fsprogs-1.46.5

e2fsprogs pakken inneholder verktøyene for å håndtere `ext2` filsystem. Det støtter også `ext3` og `ext4` journalførende filsystemer.

Omtrentlig byggetid: 4.4 SBU pÃ¥ en spinnende harddisk, 1.3 SBU pÃ¥ en SSD
Nødvendig diskplass: 93 MB

8.76.1. Installasjon av E2fsprogs

e2fsprogs dokumentasjonen anbefaler at pakken bygges i en undermappe til kildetreet:

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

Forbered e2fsprogs for kompilering:

```
../configure --prefix=/usr           \
             --sysconfdir=/etc       \
             --enable-elf-shlibs     \
             --disable-libblkid      \
             --disable-libuuid       \
             --disable-uuidd         \
             --disable-fsck
```

Betydningen av konfigureringsalternativene:

`--enable-elf-shlibs`
Dette oppretter de delte bibliotekene som noen programmer i denne pakken bruk.

`--disable-*`
Dette forhindrer e2fsprogs fra å bygge og installere `libuuid` og `libblkid` bibliotekene, `uuidd` nissen, og **fsck** innpakningen, siden util-linux installerer nyere versjoner.

Kompiler pakken:

```
make
```

For å kjøre testene, utsted:

```
make check
```

En test, u_direct_io, er kjent for å mislykkes på noen systemer.

Installer pakken:

```
make install
```

Fjern ubrukelige statiske biblioteker:

```
rm -fv /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Denne pakken installerer en gzipped `.info` fil, men oppdaterer ikke systemets `dir` fil. Pakk ut denne filen og oppdater deretter systemets `dir` fil ved hjelp av følgende kommandoer:

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir /usr/share/info/libext2fs.info
```

Hvis ønskelig, opprett og installer litt tilleggsdokumentasjon ved å utstede følgende kommandoer:

```
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
```

8.76.2. Innhold i E2fsprogs

Installerte programmer: 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, og tune2fs

Installerte biblioteker: libcom_err.so, libe2p.so, libext2fs.so, og libss.so

Installerte kataloger: /usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/ss, /usr/lib/e2fsprogs, /usr/share/et, og /usr/share/ss

Korte beskrivelser

badblocks Søker en enhet (vanligvis en diskpartisjon) etter dårlige blokker

chattr Endrer attributtene til filer på et `ext2` filsystem; det endrer også `ext3` filsystemer, journalversjonen av `ext2` filsystemet

compile_et En feiltabellkompilator; den konverterer en tabell med navn på feilkoder og meldinger til en C-kildefil som er egnet for bruk med `com_err` biblioteket

debugfs En feilsøker for filsystemet; den kan brukes til å undersøke og endre tilstanden til et `ext2` filsystem

dumpe2fs Skriver ut superblokken og informasjonen for gruppeblokkerer for filsystemet på en gitt enhet

e2freefrag Rapporterer informasjon om fragmentering av ledig plass

e2fsck Brukes til å sjekke, og eventuelt reparere `ext2` filsystemer og `ext3` filsystemer

e2image Brukes til å lagre kritiske `ext2` filsystemdata til en fil

e2label Viser eller endrer filsystemetiketten på et `ext2` filsystem tilstede på en gitt enhet

e2mmpstatus Sjekker MMP status for et ext4 filsystem

e2scrub Sjekker innholdet i et montert ext[234] filsystem

e2scrub_all Sjekker alle monterte ext[234] filsystemer for feil

e2undo Spiller angreloggen undo_log for et ext2/ext3/ext4 filsystem funnet på en enhet [Dette kan brukes til å angre en mislykket operasjon av et e2fsprogs program.]

e4crypt Ext4 filsystem krypteringsverktøy

e4defrag Tilkoblet defragmentering for ext4 filsystemer

filefrag Rapporter om hvor dårlig fragmentert en bestemt fil kan være

fsck.ext2 Som standard sjekker `ext2` filsystemer og er en hard kobling til **e2fsck**

fsck.ext3 Som standard sjekker `ext3` filsystemer og er en hard kobling til **e2fsck**

fsck.ext4 Som standard sjekker `ext4` filsystemer og er en hard kobling til **e2fsck**

logsave Lagrer utdata fra en kommando til en loggfil

lsattr Viser attributtene til filene på et andre utvidet filsystem

mk_cmds	Konverterer en tabell med kommandonavn og hjelpemeldinger til en C kildefil egnet for bruk med `libss` delsystembibliotek
mke2fs	Oppretter et `ext2` eller `ext3` filsystemet på den angitte enheten
mkfs.ext2	Som standard oppretter `ext2` filsystemer og er en hard kobling til **mke2fs**
mkfs.ext3	Som standard oppretter `ext3` filsystemer og er en hard kobling til **mke2fs**
mkfs.ext4	Som standard oppretter `ext4` filsystemer og er en hard kobling til **mke2fs**
mklost+found	Brukes til å lage en `lost+found` mappe på et `ext2` filsystem ; den forhåndstildeler diskblokker til denne katalogen for å lette oppgaven til **e2fsck**
resize2fs	Kan brukes til å forstørre eller krympe et `ext2` filsystem
tune2fs	Justerer justerbare filsystemparametere på et `ext2` filsystem
`libcom_err`	Den vanlige feilvisningsrutinen
`libe2p`	Brukt av **dumpe2fs**, **chattr**, og **lsattr**
`libext2fs`	Inneholder rutiner for å gjøre det mulig for programmer på brukernivå å manipulere et `ext2` filsystem
`libss`	Brukt av **debugfs**

8.77. Om feilsøking av symboler

De fleste programmer og biblioteker er som standard kompilert med feilsøkingssymboler inkludert (med **gcc**'s `-g` alternativ). Dette betyr at når du feilsøker et program eller bibliotek som ble kompilert med feilsøkingsinformasjon, kan feilsøkeren ikke bare gi minneadresser, men også navnene på rutinene og variablene.

Inkludering av disse feilsøkingssymbolene forstørrer imidlertid et program eller bibliotek betydelig. Følgende er et eksempel på hvor mye plass disse symbolene opptar:

- A **bash** binær med feilsøkingssymboler: 1200 KB
- A **bash** binær uten feilsøkingssymboler: 480 KB
- Glibc og GCC files (`/lib` og `/usr/lib`) med feilsøkings symboler: 87 MB
- Glibc og GCC filer uten feilsøkingssymboler: 16 MB

Størrelser kan variere avhengig av hvilken kompilator og C-bibliotek som ble brukt, men når man sammenligner programmer med og uten feilsøkingssymboler vil forskjellen vanligvis være en faktor mellom to og fem.

Fordi de fleste brukere aldri vil bruke en debugger på systemprogramvaren, mye diskplass kan gjenvinnes ved å fjerne disse symbolene. Den neste delen viser hvordan du fjerner alle feilsøkingssymboler fra programmene og bibliotekene.

8.78. Stripping

Denne delen er valgfri. Hvis den tiltenkte brukeren ikke er en programmerer og ikke planlegger å gjøre noen feilsøking på systemprogramvaren, kan systemstørrelsen reduseres med omtrent 2 GB ved å fjerne feilsøkingssymbolene fra binærfiler og biblioteker. Dette medfører ingen ulempe annet enn å ikke kunne feilsøke programvaren fullstendig lenger.

De fleste som bruker kommandoene nevnt nedenfor, opplever ikke noen vanskeligheter. Det er imidlertid lett å gjøre en skrivefeil og gjør det nye systemet ubrukelig, så før du kjører **strip** kommandoer, er det en god idé å lage en sikkerhetskopiering av LFS systemet i gjeldende tilstand.

Feilsøkingssymbolene for utvalgte biblioteker er plassert i separate filer. Denne feilsøkingsinformasjonen er nødvendig hvis det kjøres regresjonstester som bruker *valgrind* eller *gdb* senere i BLFS.

Noter at **strip** vil overskrive binær eller bibliotek filen den behandler. Dette kan krasje prosessene som bruker kode eller data fra filen. Hvis prosessen som kjører **strip** selv er påvirket, kan binærfilen eller biblioteket som blir strippet bli ødelagt og kan gjøre systemet helt ubrukelig. For å unngå det, kopierer vi noen biblioteker og binærfiler inn i `/tmp`, stripper dem der, og installer dem tilbake med **install** kommandoen. Les den relaterte oppføringen i Section 8.2.1, “Oppgraderingsproblemer” for begrunnelsen for å bruke **install** kommandoen her.

```
             libstdc++.so.6.0.29
             libitm.so.1.0.0
             libatomic.so.1.2.0"

cd /usr/lib

for LIB in $save_usrlib; do
    objcopy --only-keep-debug $LIB $LIB.dbg
    cp $LIB /tmp/$LIB
    strip --strip-unneeded /tmp/$LIB
    objcopy --add-gnu-debuglink=$LIB.dbg /tmp/$LIB
    install -vm755 /tmp/$LIB /usr/lib
    rm /tmp/$LIB
done

online_usrbin="bash find strip"
online_usrlib="libbfd-2.38.so
               libhistory.so.8.1
               libncursesw.so.6.3
               libm.so.6
               libreadline.so.8.1
               libz.so.1.2.11
               $(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
```

Et stort antall filer vil bli rapportert som at filformatet ikke er gjenkjent. Disse advarslene kan trygt ignoreres. De indikerer at disse filene er skript i stedet for binære filer.

8.79. Rydde opp

Til slutt, ryd opp i noen ekstra filer som er igjen etter å ha kjørt testene:

```
rm -rf /tmp/*
```

Det er også flere filer installert i /usr/lib og /usr/libexec mappene med filtypen .la. Disse er "libtool arkivfiler" . Som allerede sagt, er de bare nyttige når du kobler til statiske biblioteker. De er unødvendige, og potensielt skadelige, når du bruker dynamiske delte biblioteker, spesielt når du også bruker byggesystemer som ikke bruker autoverktøy. For å fjerne dem, kjør:

```
find /usr/lib /usr/libexec -name \*.la -delete
```

For mer informasjon om libtool arkivfiler, se *BLFS delen "Om Libtool Arkiver (.la) filer"*.

Kompilatoren bygd i Kapittel 6 og Kapittel 7 er fortsatt delvis installert og ikke nødvendig lenger. Fjern den med:

```
find /usr -depth -name $(uname -m)-lfs-linux-gnu\* | xargs rm -rf
```

Til slutt fjerner du den midlertidige "tester" brukerkontoen som ble opprettet på begynnelsen av forrige kapittel.

```
userdel -r tester
```

Chapter 9. Systemkonfigurasjon

9.1. Introduksjon

Dette kapittelet diskuterer konfigurasjonsfiler og systemd tjenester. Først er de generelle konfigurasjonsfilene som trengs for å sette opp nettverk presentert.

- Section 9.2, “Generell nettverkskonfigurasjon.”
- Section 9.2.3, “Konfigurerer systemvertsnavnet.”
- Section 9.2.4, “Tilpasse /etc/hosts filen.”

For det andre er problemer som påvirker riktig oppsett av enheter diskutert.

- Section 9.3, “Oversikt over enhets- og modulhåndtering.”
- Section 9.4, “Administrere enheter.”

For det tredje vises konfigurering av systemklokke og tastaturoppsett.

- Section 9.5, “Konfigurering av systemklokken.”
- Section 9.6, “Konfigurering av Linux konsollen.”

For det fjerde, en kort introduksjon til skriptene og konfigurasjons filer som brukes når brukeren logger på systemet, presenteres.

- Section 9.7, “Konfigurere systemlokaliteten.”
- Section 9.8, “Opprette /etc/inputrc filen.”

Og til slutt diskuteres konfigurering av systemd oppførsel.

- Section 9.10, “Systemd bruk og konfigurasjon.”

9.2. Generell nettverkskonfigurasjon

Denne delen gjelder kun hvis et nettverkskort skal konfigureres.

9.2.1. Konfigurasjonsfiler for nettverksgrensesnitt

Fra og med versjon 209, sender systemd en nettverkskonfigurasjons nisse (daemon) kalt **systemd-networkd** som kan brukes til grunnleggende nettverkskonfigurasjon. I tillegg, siden versjon 213, DNS navneoppløsning kan håndteres av **systemd-resolved** i stedet for den statiske `/etc/resolv.conf` filen. Begge tjenestene er aktivert som standard.

Konfigurasjonsfiler for **systemd-networkd** (og **systemd-resolved**) kan plasseres i `/usr/lib/systemd/network` eller `/etc/systemd/network`. Filer i `/etc/systemd/network` ha en høyere prioritet enn de i `/usr/lib/systemd/network`. Det finnes tre typer konfigurasjonsfiler: `.link`, `.netdev` og `.network` filene. For detaljert beskrivelser og eksempelinnhold i disse konfigurasjonsfilene, se `systemd-link(5)`, `systemd-netdev(5)` og `systemd-network(5)` manual sidene.

9.2.1.1. Navngivning av nettverksenheter

Udev tildeler normalt nettverkskortgrensesnittnavn basert på fysiske systemegenskaper som enp2s1. Hvis du ikke er sikker på hva grensesnittnavnet ditt er, du kan alltid kjøre **ip link** etter at du har startet opp systemet.

Note

Grensesnittnavnene avhenger av implementeringen og konfigurasjon av udev nissen som kjører på systemet. Udev nissen for LFS (**systemd-udevd**, installert i Section 8.71, “Systemd-250”) vil ikke kjøre med mindre LFS systemet er startet opp. Så det er upålitelig å bestemme grensesnittetnavnet som brukes i LFS systemet ved å kjøre disse kommandoene på vertens distro, *selv om du er i chroot miljøet*.

For de fleste systemer er det kun ett nettverksgrensesnitt for hver type tilkobling. For eksempel det klassiske grensesnittnavnet på en kablet tilkobling er eth0. En trådløs tilkobling vil vanligvis ha navnet wifi0 eller wlan0.

Hvis du foretrekker å bruke de klassiske eller tilpassede nettverksgrensesnittnavnene, det er tre alternative måter å gjøre det på:

- Masker udev .link filen for standardregler:

```
ln -s /dev/null /etc/systemd/network/99-default.link
```

- Lag et manuelt navneskjema, for eksempel ved å gi navn til grensesnitt med noe som "internet0", "dmz0" eller "lan0". For å gjøre det, lag .link filer i /etc/systemd/network/ som velger et eksplisitt navn eller et bedre navneskjema for ditt nettverksgrensesnitt. For eksempel:

```
cat > /etc/systemd/network/10-ether0.link << "EOF"
[Match]
# Change the MAC address as appropriate for your network device
MACAddress=12:34:45:78:90:AB

[Link]
Name=ether0
EOF
```

  Se man siden systemd.link(5) for mer informasjon.

- I /boot/grub/grub.cfg, send alternativet net.ifnames=0 på kjernens kommandolinje.

9.2.1.2. Statisk IP konfigurasjon

Kommandoen nedenfor oppretter en grunnleggende konfigurasjonsfil for et Statisk IP oppsett (bruker både systemd-networkd og systemd-resolved):

```
cat > /etc/systemd/network/10-eth-static.network << "EOF"
[Match]
Name=<network-device-name>

[Network]
Address=192.168.0.2/24
Gateway=192.168.0.1
DNS=192.168.0.1
Domains=<Your Domain Name>
EOF
```

Flere DNS oppføringer kan legges til hvis du har mer enn én DNS server. Ikke inkluder DNS- eller Domains-oppføringer hvis du har tenkt å bruke den statiske filen `/etc/resolv.conf`.

9.2.1.3. DHCP konfigurasjon

Kommandoen nedenfor oppretter en grunnleggende konfigurasjonsfil for et IPv4 DHCP oppsett:

```
cat > /etc/systemd/network/10-eth-dhcp.network << "EOF"
[Match]
Name=<network-device-name>

[Network]
DHCP=ipv4

[DHCP]
UseDomains=true
EOF
```

9.2.2. Opprette filen /etc/resolv.conf

Hvis systemet skal kobles til Internett, vil det trenges noen midler for Domain Name Service (DNS) navneløsning for å løse Internett domenenavn til IP adresser, og omvendt. Dette kan best oppnås ved å plassere IP adressen til DNS serveren, tilgjengelig fra Internett-leverandøren eller nettverksadministratoren til `/etc/resolv.conf`.

9.2.2.1. systemd-resolved konfigurasjon

Note

Hvis du bruker metoder som er inkompatible med systemd-resolved til å konfigurere nettverksgrensesnittene dine (f.eks.: ppp, etc.), eller hvis du bruker en type lokal løsning (f.eks. bind, dnsmasq, ubundet, etc.), eller annen programvare som genererer en `/etc/resolv.conf` (eks: et **resolvconf** program annet enn det levert av systemd), **systemd-resolved** service skal ikke brukes.

For å deaktivere systemd-resolved, utfør følgende kommando:

```
systemctl disable systemd-resolved
```

Når du bruker **systemd-resolved** for DNS konfigurasjon, oppretter den filen `/run/systemd/resolve/stub-resolv.conf`. Og hvis `/etc/resolv.conf` ikke finnes, den vil bli opprettet av **systemd-resolved** som en symbolkobling til `/run/systemd/resolve/stub-resolv.conf`. Så det er unødvendig å lage en `/etc/resolv.conf` manuelt.

9.2.2.2. Statisk resolv.conf konfigurasjon

Hvis en statisk `/etc/resolv.conf` er ønsket, lag den ved å kjøre følgende kommando:

```
cat > /etc/resolv.conf << "EOF"
# Begin /etc/resolv.conf

domain <Ditt domenenavn>
nameserver <IP adressen til din primære navneserver>
nameserver <IP adressen til din sekundære navneserver>

# End /etc/resolv.conf
EOF
```

`domain` erklæringen kan utelates eller erstattet med en `search` erklæring. Se man siden for resolv.conf for flere detaljer.

Erstatt *<IP adressen til navneserveren>* med IP adressen til DNS serveren som passer best for oppsettet ditt. Det vil ofte være mer enn én oppføring (krever sekundær server for reservefunksjon). Hvis du bare trenger eller ønsker én DNS-server, fjern den andre *nameserver* linjen fra filen. IP adressen kan også være en ruter på det lokale nettverket. Et annet alternativ er å bruke Googles offentlige DNS tjeneste ved å bruke IP adressene nedenfor som navneservere.

Note

Googles offentlige IPv4 DNS adresser er *8.8.8.8* og *8.8.4.4* for IPv4, og *2001:4860:4860::8888* og *2001:4860:4860::8844* for IPv6.

9.2.3. Konfigurerer systemvertsnavnet

Under oppstartsprosessen vil filen `/etc/hostname` brukes til å etablere systemets vertsnavn.

Opprett `/etc/hostname` filen og skriv inn et vertsnavn ved å kjøre:

```
echo "<lfs>" > /etc/hostname
```

<lfs> må erstattes med navnet til datamaskinen. Ikke skriv inn det fullt kvalifiserte domenenavnet (FQDN) her. Den informasjonen skal i `/etc/hosts` filen.

9.2.4. Tilpasse /etc/hosts filen

Bestem deg for et fullt kvalifisert domenenavn (FQDN) og mulige aliaser til bruk i `/etc/hosts` filen. Hvis du bruker statisk IP adresse, må du også bestemme deg for en IP adresse. Syntaksen for en vertsfiloppføring er:

```
IP_addresse minvert.eksempel.org aliaser
```

Med mindre datamaskinen skal være synlig for Internett (dvs. det er et registrert domene og en gyldig blokk med tildelte IP adresser—de fleste brukere har ikke dette), sørg for at IP adressen er i den private nettverkets IP adresseområde. Gyldige områder er:

```
Privat nettverk Adresseområde      Normalt prefiks
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 kan være et hvilket som helst tall i området 16-31. y kan være et hvilket som helst tall i område 0-255.

En gyldig privat IP adresse kan være 192.168.1.1. En gyldig FQDN for denne IPen kan være lfs.example.org.

Selv om du ikke bruker et nettverkskort, kreves det fortsatt et gyldig FQDN. Dette er nødvendig for at visse programmer, for eksempel MTAer, skal fungere ordentlig.

Opprett `/etc/hosts` filen ved å bruke følgende kommando:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.0.2> <FQDN> <HOSTNAME> [alias1] [alias2] ...
::1       localhost ip6-localhost ip6-loopback
ff02::1   ip6-allnodes
ff02::2   ip6-allrouters

# End /etc/hosts
EOF
```

<192.168.0.2>, *<FQDN>*, og *<HOSTNAME>* verdier må bli endret for spesifikke bruksområder eller krav (hvis det er tildelt en IP adresse av en nettverks-/systemadministrator og maskinen kobles til et eksisterende nettverk). De valgfrie aliasnavnene kan utelates, og *<192.168.0.2>* linjen kan utelates hvis du bruker en tilkobling konfigurert med DHCP eller IPv6 Autoconfiguration.

::1 oppføringen er IPv6 motstykket til 127.0.0.1 og representerer IPv6 loopback-grensesnittet. 127.0.1.1 er en tilbakekoblingsoppføring reservert spesielt for FQDN.

9.3. Oversikt over enhets- og modulhåndtering

I Chapter 8, installerte vi udev pakken når systemd ble bygget. Før vi går inn i detaljer om hvordan dette fungerer, er en kort historie om tidligere metoder for å håndtere utstyr i orden.

Generelt brukte Linux systemer tradisjonelt en statisk enhetsopprettings metode, hvor mange enhetsnoder ble opprettet under `/dev` (noen ganger bokstavelig talt tusenvis av noder), uavhengig av om de tilsvarende maskinvareenhetene faktisk eksisterte. Dette ble vanligvis gjort via en **MAKEDEV** skript, som inneholder et antall anrop til **mknod** programmet med det aktuelle hoved- og mindre enhetsnumre for alle mulige enheter som kan eksistere i verden.

Ved å bruke udev metoden vil bare de enhetene som oppdages av kjernen få enhetsnoder opprettet for dem. Fordi disse enhetsnodene vil være opprettet hver gang systemet starter, vil de bli lagret på et `devtmpfs` filsystem (et virtuelt filsystem som ligger utelukkende i systemminnet). Enhetsnoder krever ikke mye plass, så minnet som brukes er ubetydelig.

9.3.1. Historie

I februar 2000 ble et nytt filsystem kalt `devfs` flettet inn i 2.3.46-kjernen og ble gjort tilgjengelig under 2.4-serien med stabile kjerner. Selv om den var til stede i selve kjernekilden, fikk denne metoden for å lage enheter dynamisk aldri overveldende støtte fra kjerneutviklere.

Hovedproblemet med tilnærmingen vedtatt av `devfs` var måten den håndterte enheten på ved oppdagelse, opprettelse og navngivning. Det siste problemet enhetsnavnet på en enhetsnode, var kanskje den mest kritiske. Det er generelt akseptert at hvis enhetsnavn kan konfigureres, så skal enhetsnavn politikken være opp til en systemadministrator, og ikke pålagt dem av en spesiell(e) utvikler(e). `devfs` filsystemet led også av kjøreforhold som var iboende i utformingen og ikke kunne fikses uten en betydelig revisjon av kjernen. Den ble lenge merket som utdatert – på grunn av manglende vedlikehold – og ble til slutt fjernet fra kjernen i juni 2006.

Med utviklingen av det ustabile 2.5 kjernetreet, senere utgitt som 2.6-serien med stabile kjerner, et nytt virtuelt filsystem kalt `sysfs` ble opprettet. Jobben til `sysfs` er å eksportere en visning av systemets maskinvarekonfigurasjon til brukerprosesser. Med dette brukersynlige representasjonen, ble muligheten for å utvikle et brukerområde erstatning for `devfs` mer realistisk.

9.3.2. Udev Implementering

9.3.2.1. Sysfs

`sysfs` filsystemet ble kort nevnt ovenfor. Man kan lure på hvordan `sysfs` vet om enhetene som finnes på et system og hvilke enhetsnumre som skal brukes for dem. Drivere som har blitt kompilert inn i kjernen, registrerer objektene deres direkte med `sysfs` (devtmpfs internt) når de oppdages av kjernen. For drivere kompilert som moduler, vil registreringen skje når modulen blir lastet. Først når `sysfs` filsystemet er montert (på /sys), data som driverne registrerer med `sysfs` er tilgjengelig for brukerområdets prosesser og til udevd for behandling (inkludert modifikasjoner av enhetens noder).

9.3.2.2. Oppretting av enhetsnode

Enhetsfiler opprettes av kjernen av `devtmpfs` filesystemet. Enhver driver som ønsker å registrere en enhetsnode vil gå gjennom `devtmpfs` (via driverkjernen) for å gjøre det. Når `devtmpfs` forekomsten er montert på `/dev`, vil enhetsnoden i utgangspunktet bli opprettet med et fast navn, tillatelser og eier.

Kort tid senere vil kjernen sende en uevent til **udevd**. Basert på reglene spesifisert i filene i `/etc/udev/rules.d`, `/usr/lib/udev/rules.d`, og `/run/udev/rules.d` mappene, **udevd** vil opprette flere symbolkoblinger til enhetsnoden, eller endre tillatelsene, eieren eller gruppen, eller endre den interne **udevd** databaseoppføring (navn) for det objektet.

Reglene i disse tre katalogene er nummererte og alle tre kataloger slås sammen. Hvis **udevd** ikke finner en regel for enheten den oppretter, vil den opprettholde tillatelsene og eierskapet som `devtmpfs` brukte i utgangspunktet.

9.3.2.3. Modul lasting

Enhetsdrivere kompilert som moduler kan ha innebygde aliaser. Aliaser er synlige i utdataene til **modinfo** programmet og er vanligvis relatert til de bussspesifikke identifikatorene til enheter støttet av en modul. For eksempel *snd-fm801* driveren støtter PCI-enheter med leverandør-ID 0x1319 og enhets-ID 0x0801, og har et alias “pci:v00001319d00000801sv*sd*bc04sc01i*”. For de fleste enheter eksporterer bussdriveren aliaset til driveren som vil håndtere enheten via `sysfs`. F.eks `/sys/bus/pci/devices/0000:00:0d.0/modalias` filen kan inneholde strengen “pci:v00001319d00000801sv00001319sd00001319bc04sc01i00”. Standardreglene som følger med udev vil forårsake **udevd** å kalle **/sbin/modprobe** med innholdet i `MODALIAS` uevent miljøvariabel (som skal være samme som innholdet i `modalias` filen i sysfs), laster dermed alle moduler hvis aliaser samsvarer med denne strengen etter jokertegn ekspansjonen.

I dette eksemplet betyr dette at i tillegg til *snd-fm801*, det foreldede (og uønskede) *forte* driveren vil bli lastet hvis den er tilgjengelig. Se nedenfor for måter lasting av uønskede drivere kan bli forhindret.

Kjernen selv er også i stand til å laste moduler for nettverks protokoller, filsystemer og NLS-støtte på forespørsel.

9.3.2.4. Håndtering av direktekoblingsbare/dynamiske enheter

Når du kobler til en enhet, for eksempel en Universal Serial Bus (USB) MP3 spiller, gjenkjenner kjernen at enheten nå er tilkoblet og genererer en hendelse. Denne hendelsen håndteres deretter av **udevd** som beskrevet ovenfor.

9.3.3. Problemer med å laste moduler og lage enheter

Det er noen mulige problemer når det kommer til å automatisk opprette enhetsnoder.

9.3.3.1. En kjernemodul lastes ikke automatisk

Udev vil bare laste en modul hvis den har et bussspesifikt alias og bussdriveren eksporterer de nødvendige aliasene til `sysfs`. I andre tilfeller bør man ordne modullasting på andre måter. Med Linux-5.16.9, udev er kjent for å laste riktig skrevne drivere for INPUT, IDE, PCI, USB, SCSI, SERIO- og FireWire-enheter.

For å finne ut om enhetsdriveren du trenger har den nødvendige støtten for udev, kjør **modinfo** med modulnavnet som argument. Prøv nå å finne enhetskatalogen under `/sys/bus` og sjekk om det er a `modalias` filer der.

Hvis `modalias` filen finnes i `sysfs`, driveren støtter enheten og kan snakke med den direkte, men har ikke aliaset, det er en feil i driveren. Last inn driveren uten hjelp fra udev og forvent at problemet blir fikset senere.

Hvis det ikke er noen `modalias` filer i den aktuelle mappen under `/sys/bus`, dette betyr at kjerneutviklerne ennå ikke har lagt til støtte for modalias for denne busstypen. Med Linux-5.16.9, dette er tilfellet med ISA busser. Forvent at dette problemet blir løst i senere kjerneversjoner.

Udev er ikke ment å laste "wrapper" drivere som f.eks *snd-pcm-oss* og ikke-maskinvaredrivere som f.eks *loop* i det hele tatt.

9.3.3.2. En kjernemodul lastes ikke automatisk, og udev er ikke beregnet på å laste den

Hvis "wrapper" modulen forbedrer bare funksjonalitet levert av en annen modul (f.eks. *snd-pcm-oss* forbedrer funksjonaliteten til *snd-pcm* ved å gjøre lydkortene tilgjengelige for OSS applikasjoner), konfigurer **modprobe** for å laste inn innpakkningen etter at udev laster den innpakket modulen. For å gjøre dette, legg til en "softdep" linje til den tilsvarende `/etc/modprobe.d/<filename>.conf` filen. For eksempel:

```
softdep snd-pcm post: snd-pcm-oss
```

Merk at "softdep" kommandoen tillater også `pre:` avhengigheter, eller en blanding av begge `pre:` og `post:` avhengigheter. Se `modprobe.d(5)` manualside for mer informasjon om "softdep" syntaks og muligheter.

9.3.3.3. Udev laster inn en uønsket modul

Enten ikke bygg modulen, eller svarteliste den i en `/etc/modprobe.d/blacklist.conf` fil som gjort med *forte* modulen i eksemplet nedenfor:

```
blacklist forte
```

Svartelistede moduler kan fortsatt lastes inn manuelt med eksplisitt **modprobe** kommando.

9.3.3.4. Udev oppretter en enhet feil, eller lager en feil symbolkobling

Dette skjer vanligvis hvis en regel uventet samsvarer med en enhet. For eksempel kan en dårlig skrevet regel matche både en SCSI-disk (som ønsket) og den tilsvarende generiske SCSI-enheten (feil) av leverandøren. Finn den krenkende regelen og gjør den mer spesifikk, ved hjelp av **udevadm info** kommandoen.

9.3.3.5. Udev regelen fungerer upålitelig

Dette kan være en annen manifestasjon av det forrige problemet. Hvis ikke, og regelen din bruker `sysfs` attributter, kan det være et problem med kjernetiming, som bør fikses i senere kjerner. Foreløpig kan du omgå det ved å lage en regel som venter på den brukte `sysfs` attributten og tilføye den til `/etc/udev/rules.d/10-wait_for_sysfs.rules` filen (opprett denne filen hvis den ikke eksisterer). Gi beskjed til LFS Utviklingsliste hvis du gjør det, og det hjelper.

9.3.3.6. Udev oppretter ikke en enhet

Videre tekst forutsetter at driveren er bygget statisk inn i kjernen eller allerede er lastet inn som en modul, og du allerede har sjekket at udev ikke oppretter en enhet med feil navn.

Udev har ingen nødvendig informasjon for å lage en enhetsnode hvis en kjernedriver ikke eksporterer dataene sine til `sysfs`. Dette er mest vanlig med tredjepartsdrivere utenfor kjernetreet. Lag en statisk enhetsnode i `/usr/lib/udev/devices` med passende hoved/under nummer (se filen `devices.txt` inne i kjernedokumentasjonen eller dokumentasjon levert av tredjeparts driverleverandør). Det statiske enhetsnoden vil bli kopiert til `/dev` av **udev**.

9.3.3.7. Rekkefølgen for enhetsnavn endres tilfeldig etter omstart

Dette skyldes det faktum at udev, etter design, håndterer uevents og laster moduler parallelt, og dermed i en uforutsigbar rekkefølge. Dette vil aldri bli "fikset". Du bør ikke stole på kjerneenhetens navn er stabile. Lag heller dine egne regler som lager symbolkoblinger med stabile navn basert på noen stabile attributter til enheten, for eksempel et serienummer eller utdata fra forskjellige *_id-verktøy installert av udev. Se Section 9.4, "Administrere enheter" og Section 9.2, "Generell nettverkskonfigurasjon" for eksempler.

9.3.4. Nyttig lesning

Ytterligere nyttig dokumentasjon er tilgjengelig på følgende nettsteder:

- En brukerromsimplementering av `devfs` *http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf*
- `sysfs` filsystemet *http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf*

9.4. Administrere enheter

9.4.1. Håndtere dupliserte enheter

Som forklart i Section 9.3, "Oversikt over enhets- og modulhåndtering", rekkefølgen i hvilke enheter med samme funksjon vises i `/dev` er i hovedsak tilfeldig. Hvis du for eksempel har et USB-webkamera og en TV-tuner, noen ganger `/dev/video0` refererer til kameraet og `/dev/video1` refererer til tuneren, og noen ganger etter en omstart endres rekkefølgen. For alle klasser av maskinvare unntatt lydkort og nettverkskort kan dette fikses ved å lage udev-regler for egendefinerte vedvarende symbolkoblinger. Tilfellet med nettverkskort dekkes separat i Section 9.2, "Generell nettverkskonfigurasjon", og lydkortkonfigurasjon kan finnes i *BLFS*.

For hver av enhetene dine som sannsynligvis vil ha dette problemet (selv om problemet ikke eksisterer i din nåværende Linux distribusjon), finn den tilhørende mappen under `/sys/class` eller `/sys/block`. For videoenheter kan dette være `/sys/class/video4linux/videoX`. Finn ut attributtene som identifiserer enheten unikt (vanligvis fungerer, leverandør- og produkt-IDer og/eller serienumre):

```
udevadm info -a -p /sys/class/video4linux/video0
```

Skriv så regler som lager symbolkoblingene, f.eks.:

```
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+="web
KERNEL=="video*", ATTRS{device}=="0x036f",  ATTRS{vendor}=="0x109e", SYMLINK+="tvt

EOF
```

Resultatet er at `/dev/video0` og `/dev/video1` enheter refererer fortsatt tilfeldig til tuneren og webkameraet (og bør derfor aldri brukes direkte), men det finnes symbolkoblinger `/dev/tvtuner` og `/dev/webcam` som alltid peker på den rette enheten.

9.5. Konfigurering av systemklokken

Denne delen diskuterer hvordan du konfigurerer **systemd-timedated** systemtjeneste, som konfigurerer systemklokken og tidssonen.

Hvis du ikke kan huske om maskinvareklokken er satt til UTC eller ikke, finn ut ved å kjøre **`hwclock --localtime --show`** kommandoen. Dette vil vise hva gjeldende tid er i henhold til maskinvarens klokke. Hvis denne tiden samsvarer med hva klokken din er, er maskinvareklokken satt til lokal tid. Hvis utdataen fra **hwclock** ikke er lokal tid, er sjansen stor for at den er satt til UTC-tid. Bekreft dette ved å legge til eller trekke fra riktig antall timer for tidssonen til tiden vist av **hwclock**. For eksempel, hvis du for øyeblikket er i MST tidssonen, som også er kjent som GMT -0700, legg til syv timer til den lokale tiden.

systemd-timedated leser `/etc/adjtime`, og avhengig av innholdet i filen, setter klokken til enten UTC eller lokal tid.

Opprett `/etc/adjtime` filen med følgende innhold hvis maskinvareklokken er satt til lokal tid:

```
cat > /etc/adjtime << "EOF"
0.0 0 0.0
0
LOCAL
EOF
```

Hvis `/etc/adjtime` ikke er tilstede ved første oppstart, **systemd-timedated** vil anta at maskinvareklokken er satt til UTC og juster filen i henhold til det.

Du kan også bruke **timedatectl** verktøyet for å fortelle **systemd-timedated** om maskinvareklokken er satt til UTC eller lokal tid:

```
timedatectl set-local-rtc 1
```

timedatectl kan også brukes til å endre systemtid og tidssone.

For å endre gjeldende systemtid, utsted:

```
timedatectl set-time YYYY-MM-DD HH:MM:SS
```

Maskinvareklokken vil også bli oppdatert tilsvarende.

For å endre gjeldende tidssone, utsted:

```
timedatectl set-timezone TIMEZONE
```

Du kan få en liste over tilgjengelige tidssoner ved å kjøre:

```
timedatectl list-timezones
```

Note

Vær oppmerksom på at **timedatectl** kommandoen ikke fungerer i chroot miljøet. Det kan bare brukes etter at LFS systemet er startet opp med systemd.

9.5.1. Nettverkstidssynkronisering

Fra og med versjon 213, sender systemd en nisse (daemon) kalt **systemd-timesyncd** som kan brukes til synkronisere systemtiden med eksterne NTP servere.

Nissen er ikke ment som en erstatning for den vel etablerte NTP nissen, men bare som klientimplementering av SNTP protokollen som kan brukes for mindre avanserte oppgaver og på ressursbegrensede systemer.

Fra og med systemd versjon 216, **systemd-timesyncd** nissen er aktivert som standard. Hvis du vil deaktivere den, utsted følgende kommando:

```
systemctl disable systemd-timesyncd
```

`/etc/systemd/timesyncd.conf` filen kan brukes til å endre NTP serveren som **systemd-timesyncd** synkroniserer med.

Vær oppmerksom på at når systemklokken er satt til lokal tid, **systemd-timesyncd** vil ikke oppdatere maskinvarens klokke.

9.6. Konfigurering av Linux konsollen

Denne delen diskuterer hvordan du konfigurerer **systemd-vconsole-setup** systemtjeneste, som konfigurerer den virtuelle konsollens font og konsolltastaturet.

systemd-vconsole-setup tjenesten leser `/etc/vconsole.conf` filen for konfigurasjons informasjon. Bestemmer hvilket tastatur og skjermskrift som skal brukes. Diverse språkspesifikke HOWTOer kan også hjelpe med dette, se *http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*. Undersøk utdataen av **localectl list-keymaps** for en liste over gyldige konsolltastaturer. Se i `/usr/share/consolefonts` mappen for gyldige skjermfonter.

`/etc/vconsole.conf` filen skal inneholde linjer av formen: VARIABLE="verdi". Følgende variabler gjenkjennes:

KEYMAP
: Denne variabelen spesifiserer nøkkeltilordningstabellen for tastaturet. Hvis deaktivert, er standard `us`.

KEYMAP_TOGGLE
: Denne variabelen kan brukes til å konfigurere et andre veksletastatur og er deaktivert som standard.

FONT
: Denne variabelen spesifiserer fonten som brukes av den virtuelle konsollen.

FONT_MAP
: Denne variabelen spesifiserer konsollkartet som skal brukes.

FONT_UNIMAP
: Denne variabelen spesifiserer Unicode fontkartet.

Et eksempel for et tysk tastatur og konsoll er gitt nedenfor:

```
cat > /etc/vconsole.conf << "EOF"
KEYMAP=de-latin1
FONT=Lat2-Terminus16
EOF
```

Du kan endre KEYMAP verdi under kjøring ved å bruke **localectl** verktøyet:

```
localectl set-keymap MAP
```

Note

Vær oppmerksom på at **localectl** kommandoen ikke fungerer i chroot miljøet. Det kan bare brukes etter at LFS systemet er startet opp med systemd.

Du kan også bruke **localectl** verktøyet med tilsvarende parametere for å endre X11 tastaturoppsett, modell, variant og alternativer:

```
localectl set-x11-keymap LAYOUT [MODEL] [VARIANT] [OPTIONS]
```

For å liste opp mulige verdier for **localectl set-x11-keymap** parametere, kjør **localectl** med parametere oppført nedenfor:

list-x11-keymap-models
: Viser kjente X11 tastaturkartleggingsmodeller.

list-x11-keymap-layouts
: Viser kjente X11 tastaturkartleggingsoppsett.

list-x11-keymap-variants
: Viser kjente X11 tastaturkartvarianter.

list-x11-keymap-options
: Viser kjente X11 tastaturtilordningsalternativer.

Note

Bruk av noen av parameterne oppført ovenfor krever XKeyboard-Config pakken fra BLFS.

9.7. Konfigurere systemlokaliteten

`/etc/locale.conf` filen nedenfor setter noen miljøvariabler som er nødvendige for morsmålsstøtte. Å sette dem ordentlig resulterer i:

- Utdataene fra programmer blir oversatt til ditt morsmål
- Riktig klassifisering av tegn i bokstaver, sifre og andre klasser. Dette er nødvendig for **bash** å akseptere ordentlig ikke-ASCII-tegn i kommandolinjer i ikke-engelske språk
- Riktig alfabetisk sorteringsrekkefølge for landet
- Den riktige standard papirstørrelsen
- Riktig formatering av penge-, tids- og datoverdier

Erstatt *<ll>* nedenfor med koden på to bokstaver for ønsket språk (f.eks., “en”) og *<CC>* med tobokstavskoden for det aktuelle land (f.eks., “GB”). *<charmap>* bør erstattes med den kanoniske tegntabellen for din valgte lokalitet. Valgfri modifikatorer som f.eks “@euro” kan også være tilstede.

Listen over alle lokaliteter som støttes av Glibc kan fås ved å kjøre følgende kommando:

```
locale -a
```

Tegntabellene kan ha en rekke aliaser, f.eks., “ISO-8859-1” er også referert til som “iso8859-1” og “iso88591”. Noen applikasjoner kan ikke håndtere de forskjellige synonymene riktig (f.eks. krever “UTF-8” er skrevet som “UTF-8”, ikke “utf8”), så det er det sikreste i de fleste tilfeller å velge det kanoniske navnet for en bestemt lokalitet. Å bestemme det kanoniske navnet, kjør følgende kommando, hvor *<locale name>* er utdataen gitt av **locale -a** til din foretrukne lokalitet (“en_GB.iso88591” i vårt eksempel).

```
LC_ALL=<locale name> locale charmap
```

For “en_GB.iso88591” lokalitet, kommandoen over vil skrive ut:

```
ISO-8859-1
```

Dette resulterer i en endelig lokaleinnstilling for “en_GB.ISO-8859-1”. Det er viktig at lokaliteten funnet ved hjelp av heuristikken ovenfor testes på forhånd før det legges til Bash oppstartsfilene:

```
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
```

Kommandoene ovenfor skal skrive ut språknavnet, tegnkodingen som brukes av lokaliteten, den lokale valutaen og prefikset for å ringe før telefonnummeret for å komme inn i landet. Hvis noen av kommandoene ovenfor mislykkes med en melding som ligner på den som vises nedenfor, betyr dette at lokaliteten din enten ikke ble installert i kapittel 8 eller at det ikke støttes av standardinstallasjonen av Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Hvis dette skjer, bør du enten installere ønsket lokalitet ved å bruke **localedef** kommando, eller vurder å velge en annen lokalitet. Ytterligere instruksjoner forutsetter at det ikke er slike feilmeldinger fra Glibc.

Noen pakker utover LFS kan også mangle støtte for din valgte lokalitet. Et eksempel er X-biblioteket (en del av X Window System), som sender ut følgende feilmelding hvis lokaliteten ikke samsvarer nøyaktig med et av tegnkart navn i interne filer:

```
Warning: locale not supported by Xlib, locale set to C
```

I flere tilfeller forventer Xlib at karakterkartet vil bli oppført i store bokstaver med kanoniske bindestreker. For eksempel "ISO-8859-1" heller enn "iso88591". Det er også mulig å finne en passende spesifikasjon ved å fjerne tegnkart delen av lokalitetsspesifikasjonen. Dette kan sjekkes ved å kjøre **locale charmap** kommandoen i begge lokaliteter. For eksempel måtte man endre "de_DE.ISO-8859-15@euro" til "de_DE@euro" for å få denne lokaliteten gjenkjent av Xlib.

Andre pakker kan også fungere feil (men kanskje ikke nødvendigvis vise eventuelle feilmeldinger) hvis lokalenavnet ikke oppfyller deres forventninger. I disse tilfellene, undersøke hvordan andre Linux distribusjoner støtter lokaliteten din kan gi noe nyttig informasjon.

Når de riktige lokale innstillingene er bestemt, oppretter du `/etc/locale.conf` filen:

```
cat > /etc/locale.conf << "EOF"
LANG=<ll>_<CC>.<charmap><@modifiers>
EOF
```

Merk at du kan endre `/etc/locale.conf` med systemd **localectl** verktøyet. For å bruke **localectl** for eksempelet ovenfor, kjør:

```
localectl set-locale LANG="<ll>_<CC>.<charmap><@modifiers>"
```

Du kan også spesifisere andre språkspesifikke miljøvariabler som f.eks som `LANG`, `LC_CTYPE`, `LC_NUMERIC` eller enhver annen miljøvariabel fra **locale** utdata. Bare skille dem med et mellomrom. Et eksempel hvor `LANG` er satt som en_US.UTF-8 men `LC_CTYPE` er satt som en_US er:

```
localectl set-locale LANG="en_US.UTF-8" LC_CTYPE="en_US"
```

Note

Vær oppmerksom på at **localectl** kommandoen fungerer ikke i chroot miljøet. Det kan bare brukes etter at LFS systemet er startet opp med systemd.

"C" (standard) og "en_US" (det anbefalte for engelske brukere i USA) lokalitetene er forskjellige. "C" bruker US-ASCII 7-biters tegnsett, og behandler byte med det høye bitsettet som ugyldige tegn. Det er derfor, f.eks **ls** kommandoen erstatter dem med spørsmålstegn i det lokalet. Også et forsøk på å sende post med slike tegn fra Mutt eller Pine resulterer i at ikke-RFC-samsvarende meldinger sendes (tegnsettet i den utgående posten er indikert som "ukjent 8-bit"). Det foreslås at du bruker "C" lokalitet kun hvis du er sikker på at du aldri vil trenge 8-bits tegn.

9.8. Opprette /etc/inputrc filen

`inputrc` filen er konfigurasjonsfilen for readline biblioteket, som gir redigeringsmuligheter mens brukeren skriver en linje fra terminalen. Det fungerer ved å oversette tastaturinndata inn i spesifikke handlinger. Readline brukes av bash og de fleste andre skall som samt mange andre applikasjoner.

De fleste trenger ikke brukerspesifikk funksjonalitet så kommandoen nedenfor skaper en global `/etc/inputrc` brukes av alle som logger på. Hvis du senere bestemmer deg for at du må overstyre standardinnstillingene på et per bruker grunnlag, kan du lage en `.inputrc` fil i brukerens hjemmemappe med de modifiserte tilordningene.

For mer informasjon om hvordan du redigerer `inputrc` filen, se **info bash** under *Readline Init File* seksjonen. **info readline** er også en god informasjonskilde.

Nedenfor er en generisk global `inputrc` sammen med kommentarer for å forklare hva de ulike alternativene gjør. Merk at kommentarer ikke kan være på den samme linjen som kommandoer. Opprett filen ved å bruke følgende kommando:

```
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 8bit 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
```

9.9. Opprette /etc/shells filen

`shells` filen inneholder en liste over påloggingsskall på systemet. Programmer bruker denne filen til å bestemme om et skall er gyldig. For hvert skall skal det være en enkelt linje tilstede, bestående av skallets bane i forhold til roten av katalogstrukturen (/).

For eksempel konsulteres denne filen av **chsh** for å avgjøre om en uprivilegert bruker kan endre påloggingsskallet for sin egen konto. Hvis kommandonavnet ikke er oppført, vil brukeren bli nektet evnen til å skifte skall.

Det er et krav for applikasjoner som f.eks GDM som ikke fyller ut ansiktsnettleseren (face browser) hvis den ikke finner `/etc/shells`, eller FTP nisser (daemons) som tradisjonelt nekter brukere tilgang med skall som ikke er inkludert i denne filen.

```
cat > /etc/shells << "EOF"
# Begin /etc/shells

/bin/sh
/bin/bash

# End /etc/shells
EOF
```

9.10. Systemd bruk og konfigurasjon

9.10.1. Grunnleggende konfigurasjon

`/etc/systemd/system.conf` filen inneholder et sett av alternativer for å kontrollere grunnleggende systemoperasjoner. Standardfilen har alle oppføringer kommentert med standardinnstillingene angitt. Denne filen er hvor loggnivået kan endres, samt noen grunnleggende logginnstillinger. Se `systemd-system.conf(5)` manualside for detaljer for hvert konfigurasjonsalternativ.

9.10.2. Deaktiverer skjermtømming ved oppstart

Normal oppførsel for systemd er å tømme skjermen på slutten av oppstartssekvensen. Hvis ønskelig, kan denne oppførselen endres ved å kjøre følgende kommando:

```
mkdir -pv /etc/systemd/system/getty@tty1.service.d

cat > /etc/systemd/system/getty@tty1.service.d/noclear.conf << EOF
[Service]
TTYVTDisallocate=no
EOF
```

Oppstartsmeldingene kan alltid gjennomgås ved å bruke **`journalctl -b`** kommandoen som `root` bruker.

9.10.3. Deaktivere tmpfs for /tmp

Som standard, `/tmp` blir opprettet som en tmpfs. Hvis dette ikke er ønsket, kan det overstyres ved å utføre følgende kommando:

```
ln -sfv /dev/null /etc/systemd/system/tmp.mount
```

Alternativt, hvis en egen partisjon for `/tmp` er ønsket, spesifiser partisjonen i en `/etc/fstab` oppføring.

Warning

Ikke lag den symbolske lenken ovenfor hvis en separat partisjon brukes til `/tmp`. Dette vil forhindre rotfilsystemet (/) fra å bli remontert r/w og gjør systemet ubrukelig ved oppstart.

9.10.4. Konfigurere automatisk filoppretting og sletting

Det er flere tjenester som oppretter eller sletter filer eller mapper:

- systemd-tmpfiles-clean.service
- systemd-tmpfiles-setup-dev.service
- systemd-tmpfiles-setup.service

Systemplasseringen for konfigurasjonsfilene er `/usr/lib/tmpfiles.d/*.conf`. Den lokale konfigurasjonsfilene er i `/etc/tmpfiles.d`. Filer i `/etc/tmpfiles.d` overstyrer filer med samme navn i `/usr/lib/tmpfiles.d`. Se `tmpfiles.d(5)` manualside for filformat detaljer.

Merk at syntaksen for `/usr/lib/tmpfiles.d/*.conf` filer kan være forvirrende. For eksempel standard sletting av filer i /tmp mappen ligger i `/usr/lib/tmpfiles.d/tmp.conf` med linjen:

```
q /tmp 1777 root root 10d
```

Typefeltet, q, diskuterer å lage et undervolum med kvoter som egentlig bare er aktuelt for btrfs filsystemer. Den refererer til type v som igjen refererer til type d (mappe). Dette skaper deretter den spesifiserte mappen hvis den ikke er til stede og justerer tillatelsene og eierskap som spesifisert. Innholdet i katalogen vil være underlagt tidsbasert opprydding hvis aldersargumentet er spesifisert.

Hvis standardparametrene ikke er ønsket, bør filen bli kopiert til `/etc/tmpfiles.d` og redigert etter ønske. For eksempel:

```
mkdir -p /etc/tmpfiles.d
cp /usr/lib/tmpfiles.d/tmp.conf /etc/tmpfiles.d
```

9.10.5. Overstyre standard tjenesteatferd

Parametrene til en enhet kan overstyres ved å opprette en mappe og en konfigurasjonsfil i `/etc/systemd/system`. For eksempel:

```
mkdir -pv /etc/systemd/system/foobar.service.d

cat > /etc/systemd/system/foobar.service.d/foobar.conf << EOF
[Service]
Restart=always
RestartSec=30
EOF
```

Se `systemd.unit(5)` manualside for mer informasjon. Etter å ha opprettet konfigurasjonsfilen, kjør **`systemctl daemon-reload`** og **`systemctl restart foobar`** for å aktivere endringene i en tjeneste.

9.10.6. Feilsøking av oppstartssekvensen

I stedet for vanlige skallskript som brukes i SysVinit eller BSD stil init systemer, bruker systemd et enhetlig format for ulike typer oppstarts filer (eller enheter). Kommandoen **systemctl** is used to aktivere, deaktivere, kontrollere tilstand og få status for enhetsfiler. Her er noen eksempler på ofte brukte kommandoer:

- **systemctl list-units -t *<service>* [--all]**: viser innlastede enhetsfiler av typen service.
- **systemctl list-units -t *<target>* [--all]**: viser innlastede enhetsfiler av typen target.
- **systemctl show -p Wants *<multi-user.target>***: viser alle enheter som er avhengige av flerbrukermålet. Mål er spesielle enhetsfiler som er analoge med kjørenivåer under SysVinit.
- **systemctl status *<servicename.service>***: viser statusen til servicename.service. .service utvidelsen kan utelates hvis det ikke finnes andre enhetsfiler med samme navn, for eksempel .socket-filer (som lager en lyttekontakt som gir lignende funksjonalitet som inetd/xinetd).

9.10.7. Arbeide med Systemd Journal

Logging på et system oppstartet med systemd håndteres med systemd-journald (som standard), i stedet for en typisk unix syslog nisse (daemon). Du kan også legge til en normal syslog nisse og la begge operere side ved siden av hverandre om ønskelig. Systemd-journald programmet lagrer journaloppføringer i et binært format i stedet for en ren tekstloggfil. Å bistå med å analysere filen, kommandoen **journalctl** er gitt. Her er noen eksempler på ofte brukte kommandoer:

- **journalctl -r**: viser alt innholdet i journal i omvendt kronologisk rekkefølge.
- **journalctl -u *UNIT***: viser journalpostene knyttet til den angitte UNIT filen.
- **journalctl -b[=ID] -r**: viser journal oppføringer siden sist vellykkede oppstart (eller for oppstarts-ID) i omvendt kronologisk rekkefølge.
- **journalctl -f**: gir lignende funksjonalitet som tale -f (follow).

9.10.8. Arbeide med kjernedumper

Kjernedumper er nyttige for å feilsøke programmer som krasjet, spesielt når en nisseprosess krasjer. På systemd oppstartede systemer kjernedumping håndteres av **systemd-coredump**. Det vil logge kjernedumpen i journalen og oppbevare selve kjernedumpen i `/var/lib/systemd/coredump`. For å hente og behandle kjernedumper, **coredumpctl** verktøy er gitt. Her er noen eksempler på ofte brukte kommandoer:

- **coredumpctl -r**: viser alle kjernedumper i omvendt kronologisk rekkefølge.
- **coredumpctl -1 info**: viser informasjonen fra siste kjernedump.
- **coredumpctl -1 debug**: laster den siste kjernedumpen inn i *GDB*.

Kjernedumper kan bruke mye diskplass. Maksimal diskplass brukt av kjernedumper kan begrenses ved å lage en konfigurasjonsfil i `/etc/systemd/coredump.conf.d`. For eksempel:

```
mkdir -pv /etc/systemd/coredump.conf.d

cat > /etc/systemd/coredump.conf.d/maxuse.conf << EOF
[Coredump]
MaxUse=5G
EOF
```

Se `systemd-coredump(8)`, `coredumpctl(1)`, og `coredump.conf.d(5)` manualsidene for mer informasjon.

9.10.9. Langvarige prosesser

Fra og med systemd-230 blir alle brukerprosesser drept når en brukerøkt er avsluttet, selv om nohup brukes, eller prosessen bruker `daemon()` eller `setsid()` funksjoner. Dette er en bevisst endring fra et historisk tillatt miljø til et mer restriktiv. Den nye atferden kan forårsake problemer hvis du er avhengig av langvarige programmer (f.eks., **screen** eller **tmux**) for å forbli aktiv etter avsluttet brukerøkt. Det er tre måter å aktivere langvarige prosesser til å være aktiv etter at en brukerøkt er avsluttet.

- *Aktiver langvarig prosess for kun utvalgte brukere*: Vanlige brukere har tillatelse til å aktivere prosessforlenging med kommandoen **loginctl enable-linger** for deres egen bruker. Systemadministratorer kan bruke den samme kommandoen med et *`user`* argument for å aktivere for en bruker. Denne brukeren kan da bruke **systemd-run** kommandoen for å starte langvarige prosesser. For eksempel: **systemd-run --scope --user /usr/bin/screen**. Hvis du aktiverer forlenging for din bruker, vil user@.service forbli selv etter at alle påloggingsøktene er lukket, og vil automatisk starte ved systemoppstart. Dette har fordelen av å eksplisitt tillate og ikke tillate prosesser å kjøre etter at brukerøkten er avsluttet, men bryter bakoverkompatibiliteten med verktøy som **nohup** og verktøy som bruker `daemon()`.
- *Aktiver systemomfattende langvarig prosess*: Du kan angi *`KillUserProcesses=no`* i `/etc/systemd/logind.conf` for å muliggjøre prosessforlenging globalt for alle brukere. Dette har fordelen av å gjøre det gamle metoden tilgjengelig for alle brukere på bekostning av eksplisitt kontroll.
- *Deaktiver ved byggetid*: Du kan deaktivere systemforlenging som standard mens du bygger systemd ved å legge til bryteren *`-Ddefault-kill-user-processes=false`* til **meson** kommandoen for systemd. Dette deaktiverer helt systemds evne til å drepe brukerprosesser under øktslutt.

Chapter 10. Gjøre LFS systemet oppstartbart

10.1. Introduksjon

Det er på tide å gjøre LFS systemet oppstartbart. Dette kapittelet diskuterer å opprette `/etc/fstab` filen, bygge en kjerne for det nye LFS systemet, og installere GRUB oppstartslasteren slik at LFS systemet kan velges for oppstart ved oppstart.

10.2. Opprette /etc/fstab filen

`/etc/fstab` filen brukes av noen programmer til bestemme hvor filsystemer skal monteres som standard, i hvilken rekkefølge, og hvilke som må kontrolleres (for integritetsfeil) før montering. Lag en ny filsystemtabell som denne:

```
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

# End /etc/fstab
EOF
```

Erstatt *<xxx>*, *<yyy>*, og *<fff>* med verdiene som passer for systemet, for eksempel, `sda2`, `sda5`, og `ext4`. For detaljer om de seks feltene i denne filen, se **man 5 fstab**.

Filsystemer med MS-DOS eller Windows opprinnelse (dvs. vfat, ntfs, smbfs, cifs, iso9660, udf) trenger et spesielt alternativ, utf8, for ikke-ASCII tegn i filnavn som skal tolkes riktig. For ikke-UTF-8-lokaliteter, verdien av `iocharset` bør settes til å være det samme som tegnsettet for lokaliteten, justert på en slik måte at kjernen forstår det. Dette fungerer hvis den relevante tegnsettdefinisjonen (funnet under Filsystemer -> Støtte for morsmål ved konfigurering av kjernen) har blitt kompilert inn i kjernen eller bygget som en modul. Imidlertid, hvis tegnsettet til lokaliteten er UTF-8, det tilsvarende alternativet `iocharset=utf8` ville gjøre at filsystemet skille mellom store og små bokstaver. For å fikse dette, bruk spesialalternativet `utf8` i stedet for `iocharset=utf8`, for UTF-8 lokaliteter. “codepage” alternativet er også nødvendig for vfat- og smbfs-filsystemer. Det bør settes til tegnsettnummeret som brukes under MS-DOS i ditt land. For eksempel, for å montere USB-flash-stasjoner, ville en ru_RU.KOI8-R bruker trenge følgende i alternativdelen av monteringslinjen i `/etc/fstab`:

```
noauto,user,quiet,showexec,codepage=866,iocharset=koi8r
```

Det tilsvarende opsjonsfragmentet for ru_RU.UTF-8 brukere er:

```
noauto,user,quiet,showexec,codepage=866,utf8
```

Merk at å bruke `iocharset` er standard for `iso8859-1` (så filsystemet skiller mellom store og små bokstaver) , og `utf8` alternativet forteller kjernen å konvertere filnavnene ved hjelp av UTF-8 slik at de kan være tolket i UTF-8 lokaliteten.

Det er også mulig å spesifisere standard kodesett og iocharset verdier for noen filsystemer under kjernekonfigurasjon. De relevante parameterne er navngitt “Standard NLS alternativ” (`CONFIG_NLS_DEFAULT`), “Standard eksternt NLS alternativ” (`CONFIG_SMB_NLS_DEFAULT`), “Standard kodeside for FAT” (`CONFIG_FAT_DEFAULT_CODEPAGE`), and “Standard iocharset for FAT” (`CONFIG_FAT_DEFAULT_IOCHARSET`). Det er ingen måte å spesifisere disse innstillingene for ntfs filsystem på kjernekompileringstidspunktet.

Det er mulig å gjøre ext3 filsystemet pålitelig på tvers av strømfeil for enkelte harddisktyper. For å gjøre dette, legg til `barrier=1` monteringsalternativet til den aktuelle oppføringen i `/etc/fstab`. For å sjekke om diskstasjonen støtter dette alternativet, kjør *hdparm* på den aktuelle diskstasjonen. For eksempel hvis:

```
hdparm -I /dev/sda | grep NCQ
```

returnerer ikke-tom utdata, støttes alternativet.

Merk: Logical Volume Management (LVM) baserte partisjoner kan ikke bruke `barrier` valget.

10.3. Linux-5.16.9

Linux pakken inneholder Linux kjernen.

Omtrentlig byggetid: 1.5 - 130.0 SBU (typisk omtrent 12 SBU)
Nødvendig diskplass: 1200 - 8800 MB (typisk omtrent 1700 MB)

10.3.1. Installasjon av kjernen

Å bygge kjernen innebærer noen få trinn—konfigurasjon, kompilering og installasjon. Les `README` filen i kjernekildetreet for alternative metoder til måten denne boken konfigurerer kjernen på.

Forbered deg på kompilering ved å kjøre følgende kommando:

```
make mrproper
```

Dette sikrer at kjernetreet er helt rent. Kjerne teamet anbefaler at denne kommandoen utstedes før hver kjernekompilering. Ikke stol på at kildetreet er rent etter utpakking.

Det er flere måter å konfigurere kjernealternativene på. Vanligvis, gjøres dette for eksempel gjennom et menydrevet grensesnitt:

```
make menuconfig
```

Betydningen av valgfrie make miljøvariabler:

`LANG=<host_LANG_value> LC_ALL=`
Dette etablerer lokalinnstillingen til den som brukes på verten. Dette kan være nødvendig for et riktig menyconfig ncurses grensesnitt linjetegning på en UTF-8 linux tekstkonsoll.

Hvis brukt, sørg for å erstatte *`<host_LANG_value>`* med verdien av `$LANG` variabel fra verten din. Du kan alternativt bruk i stedet vertens verdi av `$LC_ALL` eller `$LC_CTYPE`.

make menuconfig
Dette starter et ncurses menydrevet grensesnitt. For andre (grafiske) grensesnitt, skriv **make help**.

For generell informasjon om kjernekonfigurasjon se *https://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt*. BLFS har litt informasjon angående spesielle kjernekonfigurasjonskrav for pakker utenfor av LFS på *https://www.linuxfromscratch.org/blfs/view/stable-systemd/longindex.html#kernel-config-index*. Ytterligere informasjon om konfigurering og bygging av kjernen finner du på *http://www.kroah.com/lkn/*

Note

Et godt utgangspunkt for å sette opp kjernekonfigurasjonen er å kjøre **make defconfig**. Dette vil sette grunnleggende konfigurasjon til en god tilstand som tar din nåværende systemarkitektur i betraktning.

Sørg for å aktivere/deaktivere/stille inn følgende funksjoner, ellers kan systemet ikke fungere riktig eller starte opp i det hele tatt:

```
General setup -->
   [ ] Auditing Support [CONFIG_AUDIT]
   < > Enable kernel headers through /sys/kernel/kheaders.tar.xz [CONFIG_IKH
   [*] Control Group support [CONFIG_CGROUPS]   --->
      [*] Memory controller [CONFIG_MEMCG]
   [ ] Enable deprecated sysfs features to support old userspace tools [CONF
   [*] Configure standard kernel features (expert users) [CONFIG_EXPERT] ---
      [*] open by fhandle syscalls [CONFIG_FHANDLE]
   CPU/Task time and stats accounting --->
      [*] Pressure stall information tracking [CONFIG_PSI]
General architecture-dependent options  --->
   [*] Enable seccomp to safely compute untrusted bytecode [CONFIG_SECCOMP]
Networking support  --->
  Networking options  --->
   <*> The IPv6 protocol [CONFIG_IPV6]
Device Drivers  --->
  Firmware Drivers   --->
   [*] Export DMI identification via sysfs to userspace [CONFIG_DMIID]
  Graphics support --->
   Frame buffer Devices --->
      [*] Support for frame buffer devices ----
  Generic Driver Options  --->
   [ ] Support for uevent helper [CONFIG_UEVENT_HELPER]
   [*] Maintain a devtmpfs filesystem to mount at /dev [CONFIG_DEVTMPFS]
   Firmware Loader --->
      [ ] Enable the firmware sysfs fallback mechanism [CONFIG_FW_LOADER_USE
File systems  --->
   [*] Inotify support for userspace [CONFIG_INOTIFY_USER]
  Pseudo filesystems  --->
   [*] Tmpfs POSIX Access Control Lists [CONFIG_TMPFS_POSIX_ACL]
```

Note

Mens "IPv6-protokollen" ikke strengt tatt kreves, anbefales det sterkt av systemd utviklerne.

Note

Hvis vertsmaskinvaren din bruker UEFI og du ønsker å starte opp LFS-system med det, bør du justere noen kjernekonfigurasjon som på følgende *BLFS side*.

Begrunnelsen for de ovennevnte konfigurasjonselementene:

`Enable kernel headers through /sys/kernel/kheaders.tar.xz`
Dette vil kreve **cpio** for å bygge kjernen. **cpio** er ikke installert av LFS.

`Support for uevent helper`
Å ha dette alternativet satt kan forstyrre enhetens behandling ved bruk av Udev/Eudev.

`Maintain a devtmpfs`
Dette vil opprette automatiserte enhetsnoder som er befolket av kjerne, selv uten at Udev kjører. Udev kjører så på toppen av dette, administrere tillatelser og legge til symbolkoblinger. Denne konfigurasjonen element er nødvendig for alle brukere av Udev/Eudev.

Alternativt, **make oldconfig** er kanskje mer hensiktsmessig i noen situasjoner. Se `README` filen for mer informasjon.

Hvis ønskelig, hopp over kjernekonfigurasjonen ved å kopiere kjernens konfigurasjonsfil, `.config`, fra vertssystemet (forutsatt at den er tilgjengelig) til den utpakkede `linux-5.16.9` mappen. Derimot, vi anbefaler ikke dette alternativet. Det er ofte bedre å utforske alle konfigurasjonsmenyer og lage kjernekonfigurasjonen fra grunnen av.

Kompiler kjernebildet og modulene:

```
make
```

Hvis du bruker kjernemoduler, modulkonfigurasjon i `/etc/modprobe.d` kan være nødvendig. Informasjon knyttet til moduler og kjernekonfigurasjon er lokalisert i Section 9.3, “Oversikt over enhets- og modulhåndtering” og kjerne dokumentasjon i `linux-5.16.9/Documentation` mappen. Også, `modprobe.d(5)` kan være av interesse.

Med mindre modulstøtte er deaktivert i kjernekonfigurasjonen, installere modulene med:

```
make modules_install
```

Etter at kjernekompileringen er fullført, er flere trinn nødvendig for å fullføre installasjonen. Noen filer må kopieres til `/boot` mappen.

Caution

Hvis vertssystemet har en separat /boot partisjon, kopieres filene nedenfor dit. Den enkleste måten å gjøre det på er å binde /boot på verten (utenfor chroot) til /mnt/lfs/boot før du fortsetter. Som root bruker i *vertssytemet*:

```
mount --bind /boot /mnt/lfs/boot
```

Stien til kjernebildet kan variere avhengig av plattformen som er brukt. Filnavnet nedenfor kan endres for å passe din smak, men stammen av filnavnet skal være *vmlinuz* for å være kompatibel med det automatiske oppsettet av oppstartsprosessen beskrevet i neste avsnitt. De følgende kommandoene antar et x86 arkitektur:

```
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-5.16.9-lfs-11.1-systemd
```

`System.map` er en symbolfil for kjernen. Den kartlegger funksjonsinngangspunktene til hver funksjon i kjernens API, samt adressene til kjernedatastrukturene for kjøringen av kjernen. Den brukes som en ressurs når man undersøker kjerneproblemer. Utfør følgende kommando for å installere kartfilen:

```
cp -iv System.map /boot/System.map-5.16.9
```

Kjernens konfigurasjonsfil `.config` produsert av **make menuconfig** steget ovenfor inneholder alle konfigurasjonsvalgene for kjernen som nettopp ble kompilert. Det er en god idé å beholde denne filen for fremtidig referanse:

```
cp -iv .config /boot/config-5.16.9
```

Installer dokumentasjonen for Linux kjernen:

```
install -d /usr/share/doc/linux-5.16.9
cp -r Documentation/* /usr/share/doc/linux-5.16.9
```

Det er viktig å merke seg at filene i kjernekildens mappen ikke eies av *root*. Når en pakke pakkes ut som bruker *root* (som vi gjorde inne i chroot), filene har bruker- og gruppe-IDer for hva som helst de var på pakkerens datamaskin. Dette er vanligvis ikke et problem for enhver pakke som skal installeres fordi kildetreet blir fjernet etter installasjonen. Imidlertid er Linux kildetreet ofte beholdt i lang tid. På grunn av dette er det en sjanse at hvilken bruker-ID pakkeren brukte vil bli tildelt noen på maskinen. Den personen ville da ha skrivetilgang til kjernens kilde.

Note

I mange tilfeller må konfigurasjonen av kjernen være oppdatert for pakker som vil bli installert senere i BLFS. I motsetning til andre pakker, er det ikke nødvendig å fjerne kjernekildetreet etter at den nybygde kjernen er installert.

Hvis kjernekildetreet skal beholdes, kjør **chown -R 0:0** på `linux-5.16.9` mappen å forsikre seg om at alle filer eies av brukeren *root*.

Warning

Noe kjernedokumentasjon anbefaler å opprette en symbolkobling fra `/usr/src/linux` som peker på kjernens kildemappe. Dette er spesifikt for kjerner før 2.6-serien og *må ikke* opprettes på et LFS system, for det kan forårsake problemer for pakker du kanskje ønsker å bygge når basis LFS systemet er fullstendig.

Warning

Deklarasjonene i systemets `include` mappen (`/usr/include`) bør *alltid* være de som Glibc ble kompilert mot, det vil si de desinfiserte deklarasjonene installert i Section 5.4, “Linux-5.16.9 API Headers”. Derfor bør de *aldri* erstattes av enten de rå kjernedeklarasjonene eller andre kjernerensede deklarasjoner.

10.3.2. Konfigurere Linux modul lastnings rekkefølge

Mesteparten av tiden lastes Linux moduler automatisk, men noen ganger trenger den en bestemt retning. Programmet som laster moduler, **modprobe** eller **insmod**, bruker `/etc/modprobe.d/usb.conf` for dette formålet. Denne filen må opprettes slik at hvis USB-driverne (ehci_hcd, ohci_hcd og uhci_hcd) har blitt bygget som moduler, vil de bli lastet inn i riktig rekkefølge; ehci_hcd må lastes før ohci_hcd og uhci_hcd i rekkefølge for å unngå at det sendes ut en advarsel ved oppstart.

Opprett en ny fil `/etc/modprobe.d/usb.conf` ved å kjøre følgende:

```
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
```

10.3.3. Innhold i Linux

Installerte filer: config-5.16.9, vmlinuz-5.16.9-lfs-11.1-systemd, og System.map-5.16.9
Installerte mapper: /lib/modules, /usr/share/doc/linux-5.16.9

Korte beskrivelser

`config-5.16.9`	Inneholder alle konfigurasjonsvalgene for kjernen
`vmlinuz-5.16.9-lfs-11.1-systemd`	Motoren til Linux systemet. Når du slår på datamaskinen, er kjernen den første delen av operativsystemet som blir lastet. Den oppdager og initialiserer alle komponenter i datamaskinens maskinvare, gjør deretter disse komponentene tilgjengelige som et tre med filer til programvarer og forvandler en enkelt CPU til en multitasking-maskin med å kjøre mange programmer tilsynelatende samtidig
`System.map-5.16.9`	En liste over adresser og symboler; den kartlegger inngangspunktene og adresser til alle funksjonene og datastrukturene i kjernen

10.4. Bruke GRUB til å sette opp oppstartsprosessen

Note

Hvis systemet ditt har UEFI støtte og du ønsker å starte LFS med UEFI, bør du hoppe over denne siden og konfigurere GRUB med UEFI støtte ved å bruke instruksjonene i *BLFS siden*.

10.4.1. Introduksjon

Warning

Å konfigurere GRUB feil kan gjøre systemet ditt ubrukelig uten en alternativ oppstartsenhet som en CD-ROM eller oppstartbar USB-stasjon. Denne delen er ikke nødvendig for å starte opp LFS systemet. Du kan bare endre din nåværende oppstartslaster, f.eks. Grub-Legacy, GRUB2 eller LILO.

Sørg for at en nødoppstartsdiskett er klar til å "redde" datamaskinen hvis datamaskinen blir ubrukelig (ikke-oppstartbar). Hvis du ikke allerede har en oppstartsenhet, kan du opprette en. For at prosedyren nedenfor skal fungere, så må du hoppe til BLFS og installere **`xorriso`** fra *libisoburn* pakken.

```
cd /tmp
grub-mkrescue --output=grub-img.iso
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

10.4.2. GRUB navnekonvensjoner

GRUB bruker sin egen navnestruktur for stasjoner og partisjoner i formen *(hdn,m)*, hvor *n* er harddisknummeret og *m* er partisjonensnummeret. Harddisknummeret starter fra null, men partisjonsnummeret starter fra én for vanlige partisjoner og fem for utvidede partisjoner. Merk at dette er forskjellig fra tidligere versjoner hvor begge tallene startet fra null. For eksempel partisjon `sda1` er *(hd0,1)* for GRUB og `sdb3` er *(hd1,3)*. I motsetning til Linux anser ikke GRUB CD-ROM-stasjoner som harddisker. For eksempel hvis du bruker en CD på `hdb` og en ekstra harddisk på `hdc`, vil den andre harddisken fortsatt være *(hd1)*.

10.4.3. Sette opp konfigurasjonen

GRUB fungerer ved å skrive data til det første fysiske sporet til en hardisk. Dette området er ikke en del av noe filsystem. programmene der gir tilgang til GRUB moduler i oppstartspartisjonen. Standardplasseringen er /boot/grub/.

Plasseringen av oppstartspartisjonen er et valg av brukeren som påvirker konfigurasjonen. En anbefaling er å ha en egen liten (foreslått størrelse er 200 MB) partisjon kun for oppstartsinformasjon. På den måten hver bygging, enten LFS eller en kommersiell distro, kan få tilgang til den samme oppstartsfilen og tilgang kan gjøres fra hvilket som helst oppstartssystem. Hvis du velger å gjøre dette må du montere den separate partisjonen, flytte alle filene i nåværende `/boot` mappen (f.eks linuxkjernen du nettopp bygde i forrige seksjon) til den nye partisjonen. Du må da avmontere partisjonen og montere den på nytt som `/boot`. Hvis du gjør dette, sørg for å oppdatere `/etc/fstab`.

Bruk av gjeldende lfs partisjon vil også fungere, men konfigurasjon for flere systemer er vanskeligere.

Bruk informasjonen ovenfor, finn ut hva som er riktig designator for rotpartisjonen (eller oppstartspartisjonen, hvis en separat er brukt). For det følgende eksempelet antas det at rot (eller separat oppstart) partisjon er `sda2`.

Installer GRUB filene i `/boot/grub` og sett opp oppstartssporet:

Warning

Følgende kommando vil overskrive gjeldende oppstartslaster. Ikke kjør kommandoen hvis dette ikke er ønsket, for eksempel hvis du bruker en tredjeparts boot manager for å administrere Master Boot Record (MBR).

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

Note

Hvis systemet har blitt startet opp med UEFI, **grub-install** vil prøve å installere filer for *x86_64-efi* målet, men disse filene er ikke installert i Chapter 8. Hvis dette er tilfelle, legg til `--target i386-pc` til kommandoen ovenfor.

10.4.4. Opprette GRUB konfigurasjonsfilen

Generer `/boot/grub/grub.cfg`:

```
cat > /boot/grub/grub.cfg << "EOF"
# Begin /boot/grub/grub.cfg
set default=0
set timeout=5

insmod ext2
set root=(hd0,2)

menuentry "GNU/Linux, Linux 5.16.9-lfs-11.1-systemd" {
        linux   /boot/vmlinuz-5.16.9-lfs-11.1-systemd root=/dev/sda2 ro
}
EOF
```

Note

Fra GRUB sitt perspektiv, kjernefilene er i forhold til partisjonen som brukes. Hvis du brukte en separat /boot-partisjon, fjern /boot fra ovenstående *linux* linjen. Du må også endre *set root* linjen for å peke på oppstartspartisjonen.

GRUB er et ekstremt kraftig program og det gir en enorm antall alternativer for oppstart fra et bredt utvalg av enheter, operativ systemer og partisjonstyper. Det er også mange muligheter for tilpasning som grafiske splash-skjermer, avspilling av lyder, museinngang, etc. Detaljer om disse alternativene er utenfor rammen av disse introduksjonene.

Caution

Det er en kommando, grub-mkconfig, som kan skrive en konfigurasjonsfil automatisk. Den bruker et sett med skript i /etc/grub.d/ og vil ødelegge eventuelle tilpasninger du gjør. Disse skriptene er først og fremst designet for distribusjoner uten kilder og anbefales ikke for LFS. Hvis du installerer en kommersiell Linux distribusjon, er det en god sjanse at dette programmet skal kjøres. Sørg for å sikkerhetskopiere grub.cfg filen.

Chapter 11. Slutten

11.1. Slutten

Bra gjort! Det nye LFS systemet er installert! Vi ønsker deg mye suksess med ditt skinnende nye spesialbygde Linux-system.

Det kan være lurt å lage en `/etc/lfs-release` fil. Ved å ha denne filen, er det veldig lett for deg (og for oss hvis du trenger å be om hjelp på et tidspunkt) å finne ut hvilken LFS versjon som er installert på systemet. Opprett denne filen med å kjøre:

```
echo 11.1-systemd > /etc/lfs-release
```

To filer som beskriver det installerte systemet kan brukes av pakker som kan installeres på systemet senere, enten i binær form eller ved å bygge dem.

Den første viser statusen til ditt nytt system med hensyn til Linux Standards Base (LSB). For å lage denne filen, kjør:

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="11.1-systemd"
DISTRIB_CODENAME="<your name here>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

Den andre inneholder omtrent samme informasjon, og brukes av systemd og noen grafiske skrivebordsmiljøer. For å lage denne filen, kjør:

```
cat > /etc/os-release << "EOF"
NAME="Linux From Scratch"
VERSION="11.1-systemd"
ID=lfs
PRETTY_NAME="Linux From Scratch 11.1-systemd"
VERSION_CODENAME="<your name here>"
EOF
```

Sørg for å sette inn en form for tilpasning for feltene 'DISTRIB_CODENAME' og 'VERSION_CODENAME' for å gjøre systemet ditt unikt.

11.2. Bli regnet med

Nå som du er ferdig med boken, ønsker du å bli regnet som en LFS bruker? Gå over til *https://www.linuxfromscratch.org/cgi-bin/lfscounter.php* og registrer deg som LFS bruker ved å skrive inn navnet ditt og den første LFS versjonen du har brukt.

La oss restarte inn i LFS nå.

11.3. Omstart av systemet

Nå som all programvaren er installert, er det på tide å starte datamaskinen din på nytt. Du bør imidlertid være klar over et par ting. Systemet du har skapt i denne boken er ganske minimal, og vil mest sannsynlig ikke ha funksjonaliteten du trenger for å kunne fortsette fremover. Ved å installere noen ekstra pakker fra BLFS boken mens du fortsatt er i det nåværende chroot miljøet, kan du etterlate deg selv i en mye bedre posisjon til å fortsette når du starter på nytt i den nye LFS installasjonen. Her er noen forslag:

- En tekstnettleser som f.eks *Lynx* lar deg enkelt vise BLFS boken i en virtuell terminal, mens du bygger pakker i en annen.
- *make-ca* pakken vil tillate deg å sette opp lokale pålitelige ankersertifikater, slik at systemet kan bekrefte SSL-sertifikater levert av eksterne servere (for eksempel et nettsted som bruker HTTPS).
- *GPM* pakken vil tillate deg til å utføre kopier/lim inn handlinger i dine virtuelle terminaler.
- Installere *sudo* kan være nyttig for å bygge pakker som en ikke-rootbruker og enkelt installere de resulterende pakker i ditt nye system.
- Hvis du ønsker å få tilgang til det nye systemet fra et eksternt system i et komfortabelt GUI-miljø, installer *openssh*.
- For å gjøre det enklere å hente filer over internett, installer *wget*.
- For å koble til et trådløst tilgangspunkt for nettverk, installere *wpa_supplicant*.
- Til slutt en gjennomgang av følgende konfigurasjonsfiler er også passende på dette punktet.
  - /etc/bashrc
  - /etc/dircolors
  - /etc/fstab
  - /etc/hosts
  - /etc/inputrc
  - /etc/profile
  - /etc/resolv.conf
  - /etc/vimrc
  - /root/.bash_profile
  - /root/.bashrc

Nå som vi har sagt det, la oss gå videre til å starte opp vår skinnende nye LFS installasjon for første gang! Første gå ut av chroot-miljøet:

```
logout
```

Deretter avmonterer de virtuelle filsystemene:

```
umount -v $LFS/dev/pts
umount -v $LFS/dev
umount -v $LFS/run
umount -v $LFS/proc
umount -v $LFS/sys
```

Hvis flere partisjoner ble opprettet, avmonter den andre partisjoner før du demonterer den viktigste, slik som dette:

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

Avmonter selve LFS filsystemet:

```
umount -v $LFS
```

Start nå systemet på nytt med:

```
shutdown -r now
```

Forutsatt at GRUB oppstartslasteren ble satt opp som skissert tidligere, er menyen satt til å starte opp *LFS 11.1-systemd* automatisk.

Når omstarten er fullført, er LFS systemet klart til bruk og mer programvare kan legges til for å passe dine behov.

11.4. Hva nå?

Takk for at du leste denne LFS boken. Vi håper at du fant denne boken nyttig og har lært mer om systemets opprettelsesprosess.

Nå som LFS systemet er installert, lurer du kanskje på “Hva nå?” For å svare på det spørsmålet har vi satt sammen en liste over ressurser for deg.

- Vedlikehold

  Feil og sikkerhetsmeldinger rapporteres regelmessig for all programvare. Siden et LFS system er kompilert fra kilden, er det opp til deg å holde det ajour med slike rapporter. Det er flere nettressurser som sporer slike rapporter, hvorav noen er vist nedenfor:

  - *CERT* (Computer Emergency Response Team)

    CERT har en E-postliste som publiserer sikkerhetsvarsler vedr ulike operativsystemer og applikasjoner. Abonnementsinformasjon er tilgjengelig i *http://www.us-cert.gov/cas/signup.html*.

  - Bugtraq

    Bugtraq er en fullstendig avslørende E-postliste for datasikkerhet. Den publiserer nylig oppdagede sikkerhetsproblemer, og noen ganger potensielle fikser for dem. Abonnementsinformasjon er tilgjengelig på *http://www.securityfocus.com/archive*.

- Beyond Linux From Scratch

  Boken Beyond Linux From Scratch dekker installasjons prosedyrer for et bredt spekter av programvare utenfor omfanget av LFS boken. BLFS prosjektet ligger på *https://www.linuxfromscratch.org/blfs/view/stable-systemd/*.

- LFS Tips

  LFS tipsene er en samling pedagogiske dokumenter sendt inn av frivillige i LFS miljøet. Hintene er tilgjengelige på *https://www.linuxfromscratch.org/hints/downloads/files/*.

- E-postlister

  Det er flere LFS E-postlister du kan abonnere på hvis du har behov for hjelp, ønsker å holde deg oppdatert med den siste utviklingen, ønsker å bidra til prosjektet, med mer. Se Chapter 1 - Mailing Lists for mer informasjon.

- Linux dokumentasjonsprosjektet

  Målet med Linux dokumentasjonsprosjektet (TLDP) er å samarbeide om alle problemene med Linux dokumentasjon. TLDP funksjonene en stor samling av HOWTOer, guider og man sider. Den ligger på *http://www.tldp.org/*.

Part V. Vedlegg

Appendix A. Akronymer og begreper

ABI Binært applikasjonsgrensesnitt (Application Binary Interface)

ALFS Automatisert Linux fra bunnen (Automated Linux From Scratch)

API Applikasjonsprogrammeringsgrensesnitt (Application Programming Interface)

ASCII Amerikansk standardkode for informasjonsutveksling (American Standard Code for Information Interchange)

BIOS Grunnleggende system for inndata/utdata (Basic Input/Output System)

BLFS Utover Linux fra bunnen (Beyond Linux From Scratch)

BSD Berkeley programvaredistribusjon (Berkeley Software Distribution)

chroot endre rot (change root)

CMOS Komplementær metalloksyd halvleder (Complementary Metal Oxide Semiconductor)

COS Tjenesteklasse (Class Of Service)

CPU Sentral prosesseringsenhet (Central Processing Unit)

CRC Syklisk redundanssjekk (Cyclic Redundancy Check)

CVS System for samtidige versjoner (Concurrent Versions System)

DHCP Dynamisk vertskonfigurasjonsprotokoll (Dynamic Host Configuration Protocol)

DNS Domenenavntjeneste (Domain Name Service)

EGA Forbedret grafikkadapter (Enhanced Graphics Adapter)

ELF Kjørbart og koblingsbart format (Executable and Linkable Format)

EOF Slutt på fil (End of File)

EQN ligning (equation)

ext2 andre utvidede filsystemet (second extended file system)

ext3 tredje utvidede filsystemet (third extended file system)

ext4 fjerde utvidede filsystemet (fourth extended file system)

FAQ Ofte stilte spørsmål (Frequently Asked Questions)

FHS Standard for Filsystemhierarkiet (Filesystem Hierarchy Standard)

FIFO Først inn først ut (First-In, First Out)

FQDN Fullt kvalifisert domenenavn (Fully Qualified Domain Name)

FTP Filoverføringsprotokoll (File Transfer Protocol)

GB Gigabyte (Gigabytes)

GCC GNU kompilatorsamling (GNU Compiler Collection)

GID Gruppeidentifikator (Group Identifier)

GMT Greenwich gjennomsnittstid (Greenwich Mean Time)

HTML Hypertekst markeringsspråk (Hypertext Markup Language)

IDE Integrert drivelektronikk (Integrated Drive Electronics)

IEEE Institutt for elektriske og elektroniske ingeniører (Institute of Electrical and Electronic Engineers)
IO Inndata/utdata (Input/Output)
IP Internett protokoll (Internet Protocol)
IPC Kommunikasjon mellom prosesser (Inter-Process Communication)
IRC Internett relé nettprat (Internet Relay Chat)
ISO Internasjonal organisasjon for standardisasjon (International Organization for Standardization)
ISP Internett tjenesteleverandør (Internet Service Provider)
KB Kilobyte (Kilobytes)
LED Lysemitterende diode (Light Emitting Diode)
LFS Linux fra bunnen (Linux From Scratch)
LSB Linux standardbase (Linux Standard Base)
MB Megabyte (Megabytes)
MBR Master oppstart opptak (Master Boot Record)
MD5 Meldingssammendrag 5 (Message Digest 5)
NIC Nettverksgrensesnittkort (Network Interface Card)
NLS Støtte for morsmål (Native Language Support)
NNTP Transportprotokoll for nettverksnyheter (Network News Transport Protocol)
NPTL Innebygd POSIX trådbibliotek (Native POSIX Threading Library)
OSS Åpent lydsystem (Open Sound System)
PCH Forhåndskompilerte deklarasjonsfiler (Pre-Compiled Headers)
PCRE Perlkompatibelt regulært uttrykk (Perl Compatible Regular Expression)
PID Prosessidentifikator (Process Identifier)
PTY pseudoterminal (pseudo terminal)
QOS Tjenestekvalitet (Quality Of Service)
RAM Tilfeldig tilgangsminne (Random Access Memory)
RPC Anrop for ekstern prosedyre (Remote Procedure Call)
RTC Sanntidsklokke (Real Time Clock)
SBU Standard byggeenhet (Standard Build Unit)
SCO Santa Cruz operasjonen (The Santa Cruz Operation)
SHA1 Sikker nøkkel algoritme 1 (Secure-Hash Algorithm 1)
TLDP Linux dokumentasjonsprosjekt (The Linux Documentation Project)
TFTP Triviell filoverføringsprotokoll (Trivial File Transfer Protocol)
TLS Trådlokal lagring (Thread-Local Storage)
UID Brukeridentifikator (User Identifier)
umask maske for opprettelse av brukerfil (user file-creation mask)
USB Universell seriebuss (Universal Serial Bus)

UTC Koordinert universell tid (Coordinated Universal Time)

UUID Universell unik identifikator (Universally Unique Identifier9

VC Virtuell konsoll (Virtual Console)

VGA Videografikkmatrise (Video Graphics Array)

VT Virtuell terminal (Virtual Terminal)

Appendix B. Anerkjennelser

Vi vil takke følgende personer og organisasjoner for deres bidrag til Linux From Scratch Project.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – LFS skaper
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – LFS Administrerende Redaktør
- *Jim Gifford* <jim@linuxfromscratch.org> – CLFS Prosjekt Medleder
- *Pierre Labastie* <pierre@linuxfromscratch.org> – BLFS Redaktør og ALFS Leder
- *DJ Lucas* <dj@linuxfromscratch.org> – LFS og BLFS Redaktør
- *Ken Moffat* <ken@linuxfromscratch.org> – BLFS Redaktør
- Utallige andre personer på de ulike LFS og BLFS postlistene som bidro til å gjøre denne boken mulig ved å gi sine forslag, teste boken, og sende inn feilrapporter, instruksjoner og deres erfaringer med installasjon av ulike pakker.

Oversettere

- *Manuel Canales Esparcia* <macana@macana-es.com> – Spansk LFS oversettelsesprosjekt
- *Johan Lenglet* <johan@linuxfromscratch.org> – Fransk LFS oversettelsesprosjekt frem til 2008
- *Jean-Philippe Mengual* <jmengual@linuxfromscratch.org> – Fransk LFS oversettelsesprosjekt frem til 2008-2016
- *Julien Lepiller* <jlepiller@linuxfromscratch.org> – Fransk LFS oversettelsesprosjekt 2017-nåtid
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Portugisisk LFS oversettelsesprosjekt
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – Tysk LFS oversettelsesprosjekt
- *Anton Maisak* <info@linuxfromscratch.org.ru> – Russisk LFS oversettelsesprosjekt
- *Elena Shevcova* <helen@linuxfromscratch.org.ru> – Russisk LFS oversettelsesprosjekt

Speilvedlikeholdere

Nordamerikanske speil

- *Scott Kveton* <scott@osuosl.org> – lfs.oregonstate.edu mirror
- *William Astle* <lost@l-w.net> – ca.linuxfromscratch.org mirror
- *Eujon Sellers* <jpolen@rackspace.com> – lfs.introspeed.com mirror
- *Justin Knierim* <tim@idge.net> – lfs-matrix.net mirror

Søramerikanske speil

- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – lfsmirror.lfs-es.info mirror
- *Luis Falcon* <Luis Falcon> – torredehanoi.org mirror

Europeiske speil

- *Guido Passet* <guido@primerelay.net> – nl.linuxfromscratch.org mirror
- *Bastiaan Jacques* <baafie@planet.nl> – lfs.pagefault.net mirror

- *Sven Cranshoff* <sven.cranshoff@lineo.be> – lfs.lineo.be mirror
- Scarlet Belgium – lfs.scarlet.be mirror
- *Sebastian Faulborn* <info@aliensoft.org> – lfs.aliensoft.org mirror
- *Stuart Fox* <stuart@dontuse.ms> – lfs.dontuse.ms mirror
- *Ralf Uhlemann* <admin@realhost.de> – lfs.oss-mirror.org mirror
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – at.linuxfromscratch.org mirror
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – se.linuxfromscratch.org mirror
- *Franck* <franck@linuxpourtous.com> – lfs.linuxpourtous.com mirror
- *Philippe Baque* <baque@cict.fr> – lfs.cict.fr mirror
- *Vitaly Chekasin* <gyouja@pilgrims.ru> – lfs.pilgrims.ru mirror
- *Benjamin Heil* <kontakt@wankoo.org> – lfs.wankoo.org mirror
- *Anton Maisak* <info@linuxfromscratch.org.ru> – linuxfromscratch.org.ru mirror

Asiatiske speil

- *Satit Phermsawang* <satit@wbac.ac.th> – lfs.phayoune.org mirror
- *Shizunet Co.,Ltd.* <info@shizu-net.jp> – lfs.mirror.shizu-net.jp mirror
- *Init World* <http://www.initworld.com/> – lfs.initworld.com mirror

Australske speil

- *Jason Andrade* <jason@dstc.edu.au> – au.linuxfromscratch.org mirror

Tidligere prosjektteammedlemmer

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – LFS Bokredaktør
- Archaic <archaic@linuxfromscratch.org> – LFS teknisk skribent/redaktør, HLFS prosjektleder, BLFS redaktør, vedlikeholder av tips og oppdateringsprosjektet
- *Matthew Burgess* <matthew@linuxfromscratch.org> – LFS prosjektleder, LFS teknisk skribent/redaktør
- *Nathan Coulson* <nathan@linuxfromscratch.org> – LFS-Bootscripts vedlikeholder
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Nettsted Utvikler, FAQ vedlikeholder
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – LFS/BLFS/HLFS XML og XSL vedlikeholder
- Alex Groenewoud – LFS teknisk skribent
- Marc Heerdink
- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – LFS Teknisk skribent, LFS LiveCD vedlikeholder
- *Bryan Kadzban* <bryan@linuxfromscratch.org> – LFS Teknisk forfatter
- Mark Hymers

- Seth W. Klein – FAQ vedlikeholder
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Wiki vedlikeholder
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Nettsted Backend-Scripts vedlikeholder
- *Randy McMurchy* <randy@linuxfromscratch.org> – BLFS Prosjektleder, LFS redaktør
- *Dan Nicholson* <dnicholson@linuxfromscratch.org> – LFS og BLFS redaktør
- *Alexander E. Patrakov* <alexander@linuxfromscratch.org> – LFS Teknisk skribent, LFS Internationalization Editor, LFS Live CD vedlikeholder
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – LFS NNTP Gateway vedlikeholder
- *Douglas R. Reno* <renodr@linuxfromscratch.org> – Systemd redaktør
- *Ryan Oliver* <ryan@linuxfromscratch.org> – CLFS prosjektet Medleder
- *Greg Schafer* <gschafer@zip.com.au> – LFS teknisk skribent og Arkitekt for neste generasjons 64-biters byggemetode
- Jesse Tie-Ten-Quee – LFS teknisk skribent
- *James Robertson* <jwrober@linuxfromscratch.org> – Bugzilla Vedlikeholder
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – BLFS bok Redaktør, tips og oppdateringer prosjektleder
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – LFS teknisk Forfatter, Bugzilla vedlikeholder, LFS-Bootscripts vedlikeholder
- *Zack Winkles* <zwinkles@gmail.com> – LFS teknisk skribent

Appendix C. Avhengigheter

Hver pakke bygget i LFS er avhengig av en eller flere andre pakker for å bygges og installeres riktig. Noen pakker deltar til og med i sirkulære avhengigheter, det vil si at den første pakken avhenger av den andre i sin tur avhenger av den første. På grunn av disse avhengighetene er rekkefølgen som pakkene bygges i LFS veldig viktig. Formålet med denne siden er å dokumentere avhengighetene til hver pakke bygget i LFS.

For hver pakke som bygges er det tre, og noen ganger opptil fem typer avhengigheter oppført nedenfor. Den første viser hvilke andre pakker må være tilgjengelig for å kompilere og installere den aktuelle pakken. Den andre viser pakkene som må være tilgjengelige når noen programmer eller biblioteker fra pakken brukes under kjøring. Den tredje viser hvilke pakker, i tillegg til de på den første listen, må være tilgjengelige for å kjøre testpakkene. Den fjerde listen over avhengigheter er pakker som krever at denne pakken bygges og installeres på den endelige plasseringen før de blir bygget og installert. I de fleste tilfeller er dette fordi disse pakkene hardkoder kodebaner til binærfiler i skriptene deres. Hvis ikke dette blir bygget i en bestemt rekkefølge, kan det føre til at stier til /tools/bin/[binær] blir plassert inne i skript installert i det endelige systemet. Dette er åpenbart ikke ønskelig.

Den siste listen over avhengigheter er valgfrie pakker som ikke er adressert i LFS, men kan være nyttig for brukeren. Disse pakkene kan ha ekstra obligatoriske eller valgfrie avhengigheter. For disse avhengigheter, er den anbefalte praksisen å installere dem etter fullføring av LFS boken og gå tilbake og gjenoppbygg LFS pakken. I flere tilfeller, er reinstallasjon adressert i BLFS.

Acl

Installasjonen avhenger av: Attr, Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed, og Texinfo

Påkrevd ved kjøretid: Attr og Glibc

Testpakke avhenger av: Automake, Diffutils, Findutils, og Libtool

Må installeres før: Coreutils, Sed, Tar, og Vim

Valgfrie avhengigheter: Ingen

Attr

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Automake, Diffutils, Findutils, og Libtool

Må installeres før: Acl og Libcap

Valgfrie avhengigheter: Ingen

Autoconf

Installasjonen avhenger av: Bash, Coreutils, Grep, M4, Make, Perl, Sed, og Texinfo

Påkrevd ved kjøretid: Bash, Coreutils, Grep, M4, Make, Sed, og Texinfo

Testpakke avhenger av: Automake, Diffutils, Findutils, GCC, og Libtool

Må installeres før: Automake

Valgfrie avhengigheter: *Emacs*

Automake

Installasjonen avhenger av: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed, og Texinfo

Påkrevd ved kjøretid: Bash, Coreutils, Grep, M4, Sed, og Texinfo

Testpakke avhenger av: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, og Tar

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Bash

Installasjonen avhenger av: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc, Ncurses, og Readline

Testpakke avhenger av: Expect og Shadow

Må installeres før: Ingen

Valgfrie avhengigheter: *Xorg*

Bc

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Grep, og Make

Påkrevd ved kjøretid: Glibc, Ncurses, og Readline

Testpakke avhenger av: Gawk

Må installeres før: Linux

Valgfrie avhengigheter: Ingen

Binutils

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, File, Flex, Gawk, GCC, Glibc, Grep, Make, Perl, Sed, Texinfo, og Zlib

Påkrevd ved kjøretid: Glibc og Zlib

Testpakke avhenger av: DejaGNU og Expect

Må installeres før: Ingen

Valgfrie avhengigheter: *Elfutils*

Bison

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Diffutils, Findutils, og Flex

Må installeres før: Kbd og Tar

Valgfrie avhengigheter: *Doxygen*

Bzip2

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, og Patch

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen

Må installeres før: File

Valgfrie avhengigheter: Ingen

Check

Installasjonen avhenger av: Gawk, GCC, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Bash og Gawk

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Coreutils

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Libcap, Make, OpenSSL, Patch, Perl, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Diffutils, E2fsprogs, Findutils, Shadow, og Util-linux

Må installeres før: Bash, Diffutils, Eudev, Findutils, og Man-DB

Valgfrie avhengigheter: *Expect.pm* og *IO::Tty*

DejaGNU

Installasjonen avhenger av: Bash, Coreutils, Diffutils, Expect, GCC, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Expect og Bash

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Diffutils

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Perl

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

E2fsprogs

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Sed, Texinfo, og Util-linux

Påkrevd ved kjøretid: Glibc og Util-linux

Testpakke avhenger av: Procps-ng og Psmisc

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Eudev

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Gperf, Make, Sed, og Util-linux

Påkrevd ved kjøretid: Glibc, Kmod, Xz, Util-linux, og Zlib.

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Expat

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen

Må installeres før: Python og XML::Parser

Valgfrie avhengigheter: Ingen

Expect

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, og Tcl

Påkrevd ved kjøretid: Glibc og Tcl

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: *Tk*

File

Installasjonen avhenger av: Bash, Binutils, Bzip2, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, Xz, og Zlib

Påkrevd ved kjøretid: Glibc, Bzip2, Xz, og Zlib

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: *libseccomp*

Findutils

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Bash og Glibc

Testpakke avhenger av: DejaGNU, Diffutils, og Expect

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Flex

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed, og Texinfo

Påkrevd ved kjøretid: Bash, Glibc, og M4

Testpakke avhenger av: Bison og Gawk

Må installeres før: Binutils, IProute2, Kbd, Kmod, og Man-DB

Valgfrie avhengigheter: Ingen

Gawk

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, MPFR, Patch, Readline, Sed, og Texinfo

Påkrevd ved kjøretid: Bash, Glibc, og Mpfr

Testpakke avhenger av: Diffutils

Må installeres før: Ingen

Valgfrie avhengigheter: *libsigsegv*

GCC

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, Texinfo, og Zstd

Påkrevd ved kjøretid: Bash, Binutils, Glibc, Mpc, og Python

Testpakke avhenger av: DejaGNU, Expect, og Shadow

Må installeres før: Ingen

Valgfrie avhengigheter: *GNAT* og *ISL*

GDBM

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make, og Sed

Påkrevd ved kjøretid: Bash, Glibc, og Readline

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Gettext

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Sed, og Texinfo

Påkrevd ved kjøretid: Acl, Bash, Gcc, og Glibc

Testpakke avhenger av: Diffutils, Perl, og Tcl

Må installeres før: Automake og Bison

Valgfrie avhengigheter: Ingen

Glibc

Installasjonen avhenger av: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Linux API Headers, Make, Perl, Python, Sed, og Texinfo

Påkrevd ved kjøretid: Ingen

Testpakke avhenger av: File

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

GMP

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed, og Texinfo

Påkrevd ved kjøretid: GCC og Glibc

Testpakke avhenger av: Ingen

Må installeres før: MPFR og GCC

Valgfrie avhengigheter: Ingen

Gperf

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, og Make

Påkrevd ved kjøretid: GCC og Glibc

Testpakke avhenger av: Diffutils og Expect

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Grep

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Gawk

Må installeres før: Man-DB

Valgfrie avhengigheter: *PCRE* og *libsigsegv*

Groff

Installasjonen avhenger av: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed, og Texinfo

Påkrevd ved kjøretid: GCC, Glibc, og Perl

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Man-DB og Perl

Valgfrie avhengigheter: *ghostscript* og *Uchardet*

GRUB

Installasjonen avhenger av: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo, og Xz

Påkrevd ved kjøretid: Bash, GCC, Gettext, Glibc, Xz, og Sed.

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Gzip

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Bash og Glibc

Testpakke avhenger av: Diffutils og Less

Må installeres før: Man-DB

Valgfrie avhengigheter: Ingen

Iana-Etc

Installasjonen avhenger av: Coreutils

Påkrevd ved kjøretid: Ingen

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Perl

Valgfrie avhengigheter: Ingen

Inetutils

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo, og Zlib

Påkrevd ved kjøretid: GCC, Glibc, Ncurses, og Readline

Testpakke avhenger av: Ingen

Må installeres før: Tar

Valgfrie avhengigheter: Ingen

Intltool

Installasjonen avhenger av: Bash, Gawk, Glibc, Make, Perl, Sed, og XML::Parser

Påkrevd ved kjøretid: Autoconf, Automake, Bash, Glibc, Grep, Perl, og Sed

Testpakke avhenger av: Perl

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

IProute2

Installasjonen avhenger av: Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, Libcap, Libelf, Linux API Headers, og Zlib

Påkrevd ved kjøretid: Bash, Coreutils, Glibc, Libcap, Libelf, og Zlib

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: *Berkeley DB* og *iptables*

Jinja2

Installasjonen avhenger av: MarkupSafe og Python

Påkrevd ved kjøretid: MarkupSafe og Python

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Systemd

Valgfrie avhengigheter: Ingen

Kbd

Installasjonen avhenger av: Bash, Binutils, Bison, Check, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch, og Sed

Påkrevd ved kjøretid: Bash, Coreutils, og Glibc

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Kmod

Installasjonen avhenger av: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, OpenSSL, Pkg-config, Sed, Xz, og Zlib

Påkrevd ved kjøretid: Glibc, Xz, og Zlib

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Eudev

Valgfrie avhengigheter: Ingen

Less

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, og Sed

Påkrevd ved kjøretid: Glibc og Ncurses

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Gzip

Valgfrie avhengigheter: *PCRE*

Libcap

Installasjonen avhenger av: Attr, Bash, Binutils, Coreutils, GCC, Glibc, Perl, Make, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen

Må installeres før: IProute2 og Shadow

Valgfrie avhengigheter: *Linux-PAM*

Libelf

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, og Make

Påkrevd ved kjøretid: Glibc og Zlib

Testpakke avhenger av: Ingen

Må installeres før: IProute2 og Linux

Valgfrie avhengigheter: Ingen

Libffi

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Make, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: DejaGnu

Må installeres før: Python

Valgfrie avhengigheter: Ingen

Libpipeline

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Check

Må installeres før: Man-DB

Valgfrie avhengigheter: Ingen

Libtool

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Autoconf, Automake, Bash, Binutils, Coreutils, File, GCC, Glibc, Grep, Make, og Sed

Testpakke avhenger av: Autoconf, Automake, og Findutils

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Linux

Installasjonen avhenger av: Bash, Bc, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Libelf, Make, Ncurses, OpenSSL, Perl, og Sed

Påkrevd ved kjøretid: Ingen

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: *cpio*

Linux API Headers

Installasjonen avhenger av: Bash, Binutils, Coreutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Perl, og Sed

Påkrevd ved kjøretid: Ingen

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

M4

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Bash og Glibc

Testpakke avhenger av: Diffutils

Må installeres før: Autoconf og Bison

Valgfrie avhengigheter: *libsigsegv*

Make

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Perl og Procps-ng

Må installeres før: Ingen

Valgfrie avhengigheter: *Guile*

Man-DB

Installasjonen avhenger av: Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Sed, og Xz

Påkrevd ved kjøretid: Bash, GDBM, Groff, Glibc, Gzip, Less, Libpipeline, og Zlib

Testpakke avhenger av: Util-linux

Må installeres før: Ingen

Valgfrie avhengigheter: *libseccomp*

Man-Pages

Installasjonen avhenger av: Bash, Coreutils, og Make

Påkrevd ved kjøretid: Ingen

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

MarkupSafe

Installasjonen avhenger av: Python

Påkrevd ved kjøretid: Python

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Jinja2

Valgfrie avhengigheter: Ingen

Meson

Installasjonen avhenger av: Ninja og Python

Påkrevd ved kjøretid: Python

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Systemd

Valgfrie avhengigheter: Ingen

MPC

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc, GMP, og MPFR

Testpakke avhenger av: Ingen

Må installeres før: GCC

Valgfrie avhengigheter: Ingen

MPFR

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc og GMP

Testpakke avhenger av: Ingen

Må installeres før: Gawk og GCC

Valgfrie avhengigheter: Ingen

Ncurses

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Bash, GRUB, Inetutils, Less, Procps-ng, Psmisc, Readline, Texinfo, Util-linux, og Vim

Valgfrie avhengigheter: Ingen

Ninja

Installasjonen avhenger av: Binutils, Coreutils, GCC, og Python

Påkrevd ved kjøretid: GCC og Glibc

Testpakke avhenger av: Ingen

Må installeres før: Meson

Valgfrie avhengigheter: *Asciidoc*, *Doxygen*, *Emacs*, og *re2c*

OpenSSL

Installasjonen avhenger av: Binutils, Coreutils, GCC, Make, og Perl

Påkrevd ved kjøretid: Glibc og Perl

Testpakke avhenger av: Ingen

Må installeres før: Coreutils, Kmod, og Linux

Valgfrie avhengigheter: Ingen

Patch

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, og Sed

Påkrevd ved kjøretid: Glibc og Patch

Testpakke avhenger av: Diffutils

Må installeres før: Ingen

Valgfrie avhengigheter: *Ed*

Perl

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Groff, Make, Sed, og Zlib

Påkrevd ved kjøretid: GDBM og Glibc

Testpakke avhenger av: Iana-Etc, Less. og Procps-ng

Må installeres før: Autoconf

Valgfrie avhengigheter: *Berkeley DB*

Pkg-config

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Popt, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen

Må installeres før: Kmod

Valgfrie avhengigheter: *Glib2*

Procps-ng

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Make, og Ncurses

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: DejaGNU

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Psmisc

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, og Sed

Påkrevd ved kjøretid: Glibc og Ncurses

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Python

Installasjonen avhenger av: Bash, Binutils, Coreutils, Expat, GCC, Gdbm, Gettext, Glibc, Grep, Libffi, Make, Ncurses, OpenSSL, Sed, og Util-linux

Påkrevd ved kjøretid: Bzip2, Expat, Gdbm, Glibc, Libffi, Ncurses, OpenSSL, og Zlib

Testpakke avhenger av: GDB og Valgrind

Må installeres før: Ninja

Valgfrie avhengigheter: *Berkeley DB*, *libnsl*, *SQLite*, og *Tk*

Readline

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, og Texinfo
Påkrevd ved kjøretid: Glibc og Ncurses
Testpakke avhenger av: Ingen testpakke tilgjengelig
Må installeres før: Bash og Gawk
Valgfrie avhengigheter: Ingen

Sed

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, og Texinfo
Påkrevd ved kjøretid: Acl, Attr, og Glibc
Testpakke avhenger av: Diffutils og Gawk
Må installeres før: E2fsprogs, File, Libtool, og Shadow
Valgfrie avhengigheter: Ingen

Shadow

Installasjonen avhenger av: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Libcap, Make, og Sed
Påkrevd ved kjøretid: Glibc
Testpakke avhenger av: Ingen testpakke tilgjengelig
Må installeres før: Coreutils
Valgfrie avhengigheter: *CrackLib* og *Linux-PAM*

Sysklogd

Installasjonen avhenger av: Binutils, Coreutils, GCC, Glibc, Make, og Patch
Påkrevd ved kjøretid: Glibc
Testpakke avhenger av: Ingen testpakke tilgjengelig
Må installeres før: Ingen
Valgfrie avhengigheter: Ingen

Systemd

Installasjonen avhenger av: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Expat, Gawk, GCC, Glibc, Gperf, Grep, Jinja2, Libcap, Meson, Sed, Util-linux, og Zstd
Påkrevd ved kjøretid: Acl, Attr, Glibc, Libcap, og Util-linux
Testpakke avhenger av: Ingen
Må installeres før: Ingen
Valgfrie avhengigheter: *btrfs-progs*, *cURL*, *cryptsetup*, *docbook-xml*, *docbook-xsl-nons*, *elfutils*, *Git*, *gnu-efi*, *GnuTLS*, *iptables*, *kexec-tools*, *libfido2*, *libgcrypt*, *libidn2*, *Libmicrohttpd*, *libpwquality*, *libseccomp*, *libxkbcommon*, *libxslt*, *Linux-PAM*, *lxml*, *LZ4*, *make-ca*, *p11-kit*, *PCRE2*, *Polkit*, *qemu*, *qrencode*, *quota-tools*, *rsync*, *Sphinx*, *tpm2-tss*, *Valgrind*, og *zsh*

Sysvinit

Installasjonen avhenger av: Binutils, Coreutils, GCC, Glibc, Make, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Tar

Installasjonen avhenger av: Acl, Attr, Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Acl, Attr, Bzip2, Glibc, Gzip, og Xz

Testpakke avhenger av: Autoconf, Diffutils, Findutils, Gawk, og Gzip

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Tcl

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, og Sed

Påkrevd ved kjøretid: Glibc og Zlib

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Texinfo

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, og Sed

Påkrevd ved kjøretid: Glibc og Ncurses

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Util-linux

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Eudev, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Libcap, Make, Ncurses, Sed, og Zlib

Påkrevd ved kjøretid: Glibc, Libcap, Ncurses, Readline, og Zlib

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: *Linux-PAM* og *smartmontools*

Vim

Installasjonen avhenger av: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, og Sed
Påkrevd ved kjøretid: Acl, Attr, Glibc, Python, Ncurses, og Tcl
Testpakke avhenger av: Ingen
Må installeres før: Ingen
Valgfrie avhengigheter: *Xorg*, *GTK+2*, *LessTif*, *Ruby*, og *GPM*

XML::Parser

Installasjonen avhenger av: Bash, Binutils, Coreutils, Expat, GCC, Glibc, Make, og Perl
Påkrevd ved kjøretid: Expat, Glibc, og Perl
Testpakke avhenger av: Perl
Må installeres før: Intltool
Valgfrie avhengigheter: Ingen

Xz

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, og Make
Påkrevd ved kjøretid: Glibc
Testpakke avhenger av: Ingen
Må installeres før: Eudev, File, GRUB, Kmod, og Man-DB
Valgfrie avhengigheter: Ingen

Zlib

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Make, og Sed
Påkrevd ved kjøretid: Glibc
Testpakke avhenger av: Ingen
Må installeres før: File, Kmod, Perl, og Util-linux
Valgfrie avhengigheter: Ingen

Zstd

Installasjonen avhenger av: Binutils, Coreutils, GCC, Glibc, Gzip, Make, og Xz
Påkrevd ved kjøretid: Glibc
Testpakke avhenger av: Ingen
Må installeres før: GCC og Systemd
Valgfrie avhengigheter: *LZ4*

Appendix D. LFS lisenser

Denne boken er lisensiert under Creative Commons Attribution-NonCommercial-ShareAlike 2.0 Lisensen.

Datainstruksjoner kan trekkes ut fra boken under MIT Lisensen.

D.1. Creative Commons License

Creative Commons Legal Code

Attribution-NonCommercial-ShareAlike 2.0

Important

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions
   a. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
   b. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
   c. "Licensor" means the individual or entity that offers the Work under the terms of this License.
   d. "Original Author" means the individual or entity who created the Work.
   e. "Work" means the copyrightable work of authorship offered under the terms of this License.
   f. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

g. "License Elements" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, Noncommercial, ShareAlike.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

   a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;

   b. to create and reproduce Derivative Works;

   c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

   d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works;

   The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Sections 4(e) and 4(f).

4. Restrictions.The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

   a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any reference to such Licensor or the Original Author, as requested.

   b. You may distribute, publicly display, publicly perform, or publicly digitally perform a Derivative Work only under the terms of this License, a later version of this License with the same License Elements as this License, or a Creative Commons iCommons license that contains the same License Elements as this License (e.g. Attribution-NonCommercial-ShareAlike 2.0 Japan). You must include a copy of, or the Uniform Resource Identifier for, this License or other license specified in the previous sentence with every copy or phonorecord of each Derivative Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Derivative Works that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder, and You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Derivative Work with any technological measures that control access or use of the Work in a manner

inconsistent with the terms of this License Agreement. The above applies to the Derivative Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Derivative Work itself to be made subject to the terms of this License.

c. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

d. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Derivative Works or Collective Works, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

e. For the avoidance of doubt, where the Work is a musical composition:

   i. Performance Royalties Under Blanket Licenses. Licensor reserves the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work if that performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

   ii. Mechanical Rights and Statutory Royalties. Licensor reserves the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions), if Your distribution of such cover version is primarily intended for or directed toward commercial advantage or private monetary compensation. 6. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

f. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

   a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

   b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

   a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

   b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

   c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

   d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

   e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Important

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, neither party will use the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time.

Creative Commons may be contacted at *http://creativecommons.org/*.

D.2. The MIT License

Index

Packages

Acl: 133
Attr: 132
Autoconf: 168
Automake: 170
Bash: 154
  tools: 59
Bash: 154
  tools: 59
Bc: 119
Binutils: 125
  tools, pass 1: 44
  tools, pass 2: 72
Binutils: 125
  tools, pass 1: 44
  tools, pass 2: 72
Binutils: 125
  tools, pass 1: 44
  tools, pass 2: 72
Bison: 152
  tools: 85
Bison: 152
  tools: 85
Bzip2: 110
Check: 187
Coreutils: 182
  tools: 60
Coreutils: 182
  tools: 60
D-Bus: 218
DejaGNU: 124
Diffutils: 188
  tools: 61
Diffutils: 188
  tools: 61
E2fsprogs: 230
Expat: 159
Expect: 123
File: 115
  tools: 62
File: 115
  tools: 62
Findutils: 190
  tools: 63
Findutils: 190
  tools: 63
Flex: 120
Gawk: 189
  tools: 64
Gawk: 189
  tools: 64
GCC: 139
  tools, libstdc++ pass 1: 53
  tools, libstdc++ pass 2: 82
  tools, pass 1: 46
  tools, pass 2: 73
GCC: 139
  tools, libstdc++ pass 1: 53
  tools, libstdc++ pass 2: 82
  tools, pass 1: 46
  tools, pass 2: 73
GCC: 139
  tools, libstdc++ pass 1: 53
  tools, libstdc++ pass 2: 82
  tools, pass 1: 46
  tools, pass 2: 73
GCC: 139
  tools, libstdc++ pass 1: 53
  tools, libstdc++ pass 2: 82
  tools, pass 1: 46
  tools, pass 2: 73
GCC: 139
  tools, libstdc++ pass 1: 53
  tools, libstdc++ pass 2: 82
  tools, pass 1: 46
  tools, pass 2: 73
GDBM: 157
Gettext: 150
  tools: 84
Gettext: 150
  tools: 84
Glibc: 101
  tools: 50
Glibc: 101
  tools: 50
GMP: 128
Gperf: 158
Grep: 153
  tools: 65
Grep: 153
  tools: 65

Groff: 192
GRUB: 195
Gzip: 197
  tools: 66
Gzip: 197
  tools: 66
Iana-Etc: 100
Inetutils: 160
Intltool: 167
IPRoute2: 198
Jinja2: 212
Kbd: 200
Kmod: 173
Less: 162
Libcap: 134
Libelf: 175
libffi: 176
Libpipeline: 202
Libtool: 156
Linux: 256
  tools, API headers: 49
Linux: 256
  tools, API headers: 49
M4: 118
  tools: 56
M4: 118
  tools: 56
Make: 203
  tools: 67
Make: 203
  tools: 67
Man-DB: 220
Man-pages: 99
MarkupSafe: 211
Meson: 181
MPC: 131
MPFR: 130
Ncurses: 145
  tools: 57
Ncurses: 145
  tools: 57
Ninja: 179
OpenSSL: 171
Patch: 204
  tools: 68
Patch: 204
  tools: 68
Perl: 163
  tools: 86
Perl: 163
  tools: 86
Pkgconfig: 144
Procps-ng: 223
Psmisc: 149
Python: 177
  temporary: 87
Python: 177
  temporary: 87
Readline: 116
Sed: 148
  tools: 69
Sed: 148
  tools: 69
Shadow: 135
  configuring: 136
Shadow: 135
  configuring: 136
systemd: 213
Tar: 205
  tools: 70
Tar: 205
  tools: 70
Tcl: 121
Texinfo: 206
  temporary: 88
Texinfo: 206
  temporary: 88
Udev
  usage: 240
Util-linux: 225
  tools: 89
Util-linux: 225
  tools: 89
Vim: 208
XML::Parser: 166
Xz: 112
  tools: 71
Xz: 112
  tools: 71
Zlib: 109
zstd: 114

Programs

[: 182, 183

2to3: 177
accessdb: 220, 221
aclocal: 170, 170
aclocal-1.16: 170, 170
addftinfo: 192, 192
addpart: 225, 226
addr2line: 125, 126
afmtodit: 192, 192
agetty: 225, 226
apropos: 220, 222
ar: 125, 127
as: 125, 127
attr: 132, 132
autoconf: 168, 168
autoheader: 168, 168
autom4te: 168, 168
automake: 170, 170
automake-1.16: 170, 170
autopoint: 150, 150
autoreconf: 168, 168
autoscan: 168, 168
autoupdate: 168, 168
awk: 189, 189
b2sum: 182, 183
badblocks: 230, 231
base64: 182, 183, 182, 183
base64: 182, 183, 182, 183
basename: 182, 183
basenc: 182, 183
bash: 154, 155
bashbug: 154, 155
bc: 119, 119
bison: 152, 152
blkdiscard: 225, 226
blkid: 225, 226
blkzone: 225, 226
blockdev: 225, 226
bridge: 198, 198
bunzip2: 110, 111
busctl: 213, 215
bzcat: 110, 111
bzcmp: 110, 111
bzdiff: 110, 111
bzegrep: 110, 111
bzfgrep: 110, 111
bzgrep: 110, 111
bzip2: 110, 111
bzip2recover: 110, 111
bzless: 110, 111
bzmore: 110, 111
c++: 139, 142
c++filt: 125, 127
cal: 225, 226
capsh: 134, 134
captoinfo: 145, 146
cat: 182, 183
catman: 220, 222
cc: 139, 143
cfdisk: 225, 226
chacl: 133, 133
chage: 135, 137
chattr: 230, 231
chcon: 182, 184
chcpu: 225, 226
checkmk: 187, 187
chem: 192, 192
chfn: 135, 137
chgpasswd: 135, 137
chgrp: 182, 184
chmem: 225, 226
chmod: 182, 184
choom: 225, 226
chown: 182, 184
chpasswd: 135, 137
chroot: 182, 184
chrt: 225, 226
chsh: 135, 137
chvt: 200, 201
cksum: 182, 184
clear: 145, 147
cmp: 188, 188
col: 225, 227
colcrt: 225, 227
colrm: 225, 227
column: 225, 227
comm: 182, 184
compile_et: 230, 231
coredumpctl: 213, 215
corelist: 163, 164
cp: 182, 184
cpan: 163, 164
cpp: 139, 143
csplit: 182, 184
ctrlaltdel: 225, 227

ctstat: 198, 198
cut: 182, 184
c_rehash: 171, 172
date: 182, 184
dbus-cleanup-sockets: 218, 219
dbus-daemon: 218, 219
dbus-launch: 218, 219
dbus-monitor: 218, 219
dbus-run-session: 218, 219
dbus-send: 218, 219
dbus-test-tool: 218, 219
dbus-update-activation-environment: 218, 219
dbus-uuidgen: 218, 219
dc: 119, 119
dd: 182, 184
deallocvt: 200, 201
debugfs: 230, 231
dejagnu: 124, 124
delpart: 225, 227
depmod: 173, 173
df: 182, 184
diff: 188, 188
diff3: 188, 188
dir: 182, 184
dircolors: 182, 184
dirname: 182, 184
dmesg: 225, 227
dnsdomainname: 160, 161
du: 182, 184
dumpe2fs: 230, 231
dumpkeys: 200, 201
e2freefrag: 230, 231
e2fsck: 230, 231
e2image: 230, 231
e2label: 230, 231
e2mmpstatus: 230, 231
e2scrub: 230, 231
e2scrub_all: 230, 231
e2undo: 230, 231
e4crypt: 230, 231
e4defrag: 230, 231
echo: 182, 184
egrep: 153, 153
eject: 225, 227
elfedit: 125, 127
enc2xs: 163, 164
encguess: 163, 164
env: 182, 184
envsubst: 150, 150
eqn: 192, 192
eqn2graph: 192, 192
ex: 208, 210
expand: 182, 184
expect: 123, 123
expiry: 135, 137
expr: 182, 184
factor: 182, 184
faillog: 135, 137
fallocate: 225, 227
false: 182, 184
fdisk: 225, 227
fgconsole: 200, 201
fgrep: 153, 153
file: 115, 115
filefrag: 230, 231
fincore: 225, 227
find: 190, 190
findfs: 225, 227
findmnt: 225, 227
flex: 120, 120
flex++: 120, 120
flock: 225, 227
fmt: 182, 184
fold: 182, 184
free: 223, 223
fsck: 225, 227
fsck.cramfs: 225, 227
fsck.ext2: 230, 231
fsck.ext3: 230, 231
fsck.ext4: 230, 231
fsck.minix: 225, 227
fsfreeze: 225, 227
fstrim: 225, 227
ftp: 160, 161
fuser: 149, 149
g++: 139, 143
gawk: 189, 189
gawk-5.1.1: 189, 189
gcc: 139, 143
gc-ar: 139, 143
gc-nm: 139, 143
gc-ranlib: 139, 143
gcov: 139, 143
gcov-dump: 139, 143

gcov-tool: 139, 143
gdbmtool: 157, 157
gdbm_dump: 157, 157
gdbm_load: 157, 157
gdiffmk: 192, 192
gencat: 101, 106
genl: 198, 198
getcap: 134, 134
getconf: 101, 106
getent: 101, 106
getfacl: 133, 133
getfattr: 132, 132
getkeycodes: 200, 201
getopt: 225, 227
getpcaps: 134, 134
getsubids: 135, 137
gettext: 150, 150
gettext.sh: 150, 150
gettextize: 150, 150
glilypond: 192, 192
gpasswd: 135, 137
gperf: 158, 158
gperl: 192, 192
gpinyin: 192, 192
gprof: 125, 127
grap2graph: 192, 192
grep: 153, 153
grn: 192, 193
grodvi: 192, 193
groff: 192, 193
groffer: 192, 193
grog: 192, 193
grolbp: 192, 193
grolj4: 192, 193
gropdf: 192, 193
grops: 192, 193
grotty: 192, 193
groupadd: 135, 137
groupdel: 135, 137
groupmems: 135, 137
groupmod: 135, 137
groups: 182, 184
grpck: 135, 137
grpconv: 135, 138
grpunconv: 135, 138
grub-bios-setup: 195, 196
grub-editenv: 195, 196
grub-file: 195, 196
grub-fstest: 195, 196
grub-glue-efi: 195, 196
grub-install: 195, 196
grub-kbdcomp: 195, 196
grub-macbless: 195, 196
grub-menulst2cfg: 195, 196
grub-mkconfig: 195, 196
grub-mkimage: 195, 196
grub-mklayout: 195, 196
grub-mknetdir: 195, 196
grub-mkpasswd-pbkdf2: 195, 196
grub-mkrelpath: 195, 196
grub-mkrescue: 195, 196
grub-mkstandalone: 195, 196
grub-ofpathname: 195, 196
grub-probe: 195, 196
grub-reboot: 195, 196
grub-render-label: 195, 196
grub-script-check: 195, 196
grub-set-default: 195, 196
grub-setup: 195, 196
grub-syslinux2cfg: 195, 196
gunzip: 197, 197
gzexe: 197, 197
gzip: 197, 197
h2ph: 163, 164
h2xs: 163, 164
halt: 213, 215
head: 182, 184
hexdump: 225, 227
hostid: 182, 184
hostname: 160, 161
hostnamectl: 213, 215
hpftodit: 192, 193
hwclock: 225, 227
i386: 225, 227
iconv: 101, 107
iconvconfig: 101, 107
id: 182, 184
idle3: 177
ifcfg: 198, 198
ifconfig: 160, 161
ifnames: 168, 169
ifstat: 198, 198
indxbib: 192, 193
info: 206, 207

infocmp: 145, 147
infotocap: 145, 147
init: 213, 215
insmod: 173, 174
install: 182, 184
install-info: 206, 207
instmodsh: 163, 164
intltool-extract: 167, 167
intltool-merge: 167, 167
intltool-prepare: 167, 167
intltool-update: 167, 167
intltoolize: 167, 167
ionice: 225, 227
ip: 198, 198
ipcmk: 225, 227
ipcrm: 225, 227
ipcs: 225, 227
irqtop: 225, 227
isosize: 225, 227
join: 182, 184
journalctl: 213, 215
json_pp: 163, 164
kbdinfo: 200, 201
kbdrate: 200, 201
kbd_mode: 200, 201
kernel-install: 213, 215
kill: 225, 227
killall: 149, 149
kmod: 173, 174
last: 225, 227
lastb: 225, 227
lastlog: 135, 138
ld: 125, 127
ld.bfd: 125, 127
ld.gold: 125, 127
ldattach: 225, 227
ldconfig: 101, 107
ldd: 101, 107
lddlibc4: 101, 107
less: 162, 162
lessecho: 162, 162
lesskey: 162, 162
lex: 120, 120
lexgrog: 220, 222
lfskernel-5.16.9: 256, 260
libasan: 139, 143
libatomic: 139, 143
libcc1: 139, 143
libnetcfg: 163, 164
libtool: 156, 156
libtoolize: 156, 156
link: 182, 185
linux32: 225, 228
linux64: 225, 228
lkbib: 192, 193
ln: 182, 185
lnstat: 198, 199
loadkeys: 200, 201
loadunimap: 200, 201
locale: 101, 107
localectl: 213, 215
localedef: 101, 107
locate: 190, 190
logger: 225, 228
login: 135, 138
loginctl: 213, 215
logname: 182, 185
logoutd: 135, 138
logsave: 230, 231
look: 225, 228
lookbib: 192, 193
losetup: 225, 228
ls: 182, 185
lsattr: 230, 231
lsblk: 225, 228
lscpu: 225, 228
lsipc: 225, 228
lsirq: 225, 228
lslocks: 225, 228
lslogins: 225, 228
lsmem: 225, 228
lsmod: 173, 174
lsns: 225, 228
lto-dump: 139, 143
lzcat: 112, 112
lzcmp: 112, 112
lzdiff: 112, 112
lzegrep: 112, 112
lzfgrep: 112, 112
lzgrep: 112, 112
lzless: 112, 112
lzma: 112, 112
lzmadec: 112, 113
lzmainfo: 112, 113

lzmore: 112, 113
m4: 118, 118
machinectl: 213, 215
make: 203, 203
makedb: 101, 107
makeinfo: 206, 207
man: 220, 222
man-recode: 220, 222
mandb: 220, 222
manpath: 220, 222
mapscrn: 200, 201
mcookie: 225, 228
md5sum: 182, 185
mesg: 225, 228
meson: 181, 181
mkdir: 182, 185
mke2fs: 230, 232
mkfifo: 182, 185
mkfs: 225, 228
mkfs.bfs: 225, 228
mkfs.cramfs: 225, 228
mkfs.ext2: 230, 232
mkfs.ext3: 230, 232
mkfs.ext4: 230, 232
mkfs.minix: 225, 228
mklost+found: 230, 232
mknod: 182, 185
mkswap: 225, 228
mktemp: 182, 185
mk_cmds: 230, 232
mmroff: 192, 193
modinfo: 173, 174
modprobe: 173, 174
more: 225, 228
mount: 225, 228
mountpoint: 225, 228
msgattrib: 150, 150
msgcat: 150, 151
msgcmp: 150, 151
msgcomm: 150, 151
msgconv: 150, 151
msgen: 150, 151
msgexec: 150, 151
msgfilter: 150, 151
msgfmt: 150, 151
msggrep: 150, 151
msginit: 150, 151
msgmerge: 150, 151
msgunfmt: 150, 151
msguniq: 150, 151
mtrace: 101, 107
mv: 182, 185
namei: 225, 228
ncursesw6-config: 145, 147
neqn: 192, 193
networkctl: 213, 215
newgidmap: 135, 138
newgrp: 135, 138
newuidmap: 135, 138
newusers: 135, 138
ngettext: 150, 151
nice: 182, 185
ninja: 179, 180
nl: 182, 185
nm: 125, 127
nohup: 182, 185
nologin: 135, 138
nproc: 182, 185
nroff: 192, 193
nscd: 101, 107
nsenter: 225, 228
nstat: 198, 199
numfmt: 182, 185
objcopy: 125, 127
objdump: 125, 127
od: 182, 185
oomctl: 213, 216
openssl: 171, 172
openvt: 200, 201
partx: 225, 228
passwd: 135, 138
paste: 182, 185
patch: 204, 204
pathchk: 182, 185
pcprofiledump: 101, 107
pdfmom: 192, 193
pdfroff: 192, 193
pdftexi2dvi: 206, 207
peekfd: 149, 149
perl: 163, 164
perl5.34.0: 163, 164
perlbug: 163, 164
perldoc: 163, 164
perlivp: 163, 165

perlthanks: 163, 165
pfbtops: 192, 193
pgrep: 223, 223
pic: 192, 193
pic2graph: 192, 193
piconv: 163, 165
pidof: 223, 223
ping: 160, 161
ping6: 160, 161
pinky: 182, 185
pip3: 177
pivot_root: 225, 228
pkg-config: 144, 144
pkill: 223, 224
pl2pm: 163, 165
pldd: 101, 107
pmap: 223, 224
pod2html: 163, 165
pod2man: 163, 165
pod2texi: 206, 207
pod2text: 163, 165
pod2usage: 163, 165
podchecker: 163, 165
podselect: 163, 165
portablectl: 213, 216
post-grohtml: 192, 193
poweroff: 213, 216
pr: 182, 185
pre-grohtml: 192, 193
preconv: 192, 193
printenv: 182, 185
printf: 182, 185
prlimit: 225, 228
prove: 163, 165
prtstat: 149, 149
ps: 223, 224
psfaddtable: 200, 201
psfgettable: 200, 201
psfstriptable: 200, 201
psfxtable: 200, 201
pslog: 149, 149
pstree: 149, 149
pstree.x11: 149, 149
ptar: 163, 165
ptardiff: 163, 165
ptargrep: 163, 165
ptx: 182, 185
pwait: 223, 224
pwck: 135, 138
pwconv: 135, 138
pwd: 182, 185
pwdx: 223, 224
pwunconv: 135, 138
pydoc3: 177
python3: 177
ranlib: 125, 127
readelf: 125, 127
readlink: 182, 185
readprofile: 225, 228
realpath: 182, 185
reboot: 213, 216
recode-sr-latin: 150, 151
refer: 192, 193
rename: 225, 228
renice: 225, 228
reset: 145, 147
resize2fs: 230, 232
resizepart: 225, 228
resolvconf: 213, 216
resolvectl: 213, 216
rev: 225, 228
rkfill: 225, 228
rm: 182, 185
rmdir: 182, 185
rmmod: 173, 174
roff2dvi: 192, 194
roff2html: 192, 194
roff2pdf: 192, 194
roff2ps: 192, 194
roff2text: 192, 194
roff2x: 192, 194
routef: 198, 199
routel: 198, 199
rtacct: 198, 199
rtcwake: 225, 228
rtmon: 198, 199
rtpr: 198, 199
rtstat: 198, 199
runcon: 182, 185
runlevel: 213, 216
runtest: 124, 124
rview: 208, 210
rvim: 208, 210
script: 225, 228

scriptlive: 225, 229
scriptreplay: 225, 229
sdiff: 188, 188
sed: 148, 148
seq: 182, 185
setarch: 225, 229
setcap: 134, 134
setfacl: 133, 133
setfattr: 132, 132
setfont: 200, 201
setkeycodes: 200, 201
setleds: 200, 201
setmetamode: 200, 201
setsid: 225, 229
setterm: 225, 229
setvtrgb: 200, 201
sfdisk: 225, 229
sg: 135, 138
sh: 154, 155
sha1sum: 182, 185
sha224sum: 182, 185
sha256sum: 182, 185
sha384sum: 182, 185
sha512sum: 182, 186
shasum: 163, 165
showconsolefont: 200, 201
showkey: 200, 201
shred: 182, 186
shuf: 182, 186
shutdown: 213, 216
size: 125, 127
slabtop: 223, 224
sleep: 182, 186
sln: 101, 107
soelim: 192, 194
sort: 182, 186
sotruss: 101, 107
splain: 163, 165
split: 182, 186
sprof: 101, 107
ss: 198, 199
stat: 182, 186
stdbuf: 182, 186
strings: 125, 127
strip: 125, 127
stty: 182, 186
su: 135, 138
sulogin: 225, 229
sum: 182, 186
swaplabel: 225, 229
swapoff: 225, 229
swapon: 225, 229
switch_root: 225, 229
sync: 182, 186
sysctl: 223, 224
systemctl: 213, 216
systemd-analyze: 213, 216
systemd-ask-password: 213, 216
systemd-cat: 213, 216
systemd-cgls: 213, 216
systemd-cgtop: 213, 216
systemd-creds: 213, 216
systemd-delta: 213, 216
systemd-detect-virt: 213, 216
systemd-dissect: 213, 216
systemd-escape: 213, 216
systemd-hwdb: 213, 216
systemd-id128: 213, 216
systemd-inhibit: 213, 216
systemd-machine-id-setup: 213, 217
systemd-mount: 213, 217
systemd-notify: 213, 217
systemd-nspawn: 213, 217
systemd-path: 213, 217
systemd-repart: 213, 217
systemd-resolve: 213, 217
systemd-run: 213, 217
systemd-socket-activate: 213, 217
systemd-sysext: 213, 217
systemd-tmpfiles: 213, 217
systemd-tty-ask-password-agent: 213, 217
systemd-umount: 213, 217
tabs: 145, 147
tac: 182, 186
tail: 182, 186
talk: 160, 161
tar: 205, 205
taskset: 225, 229
tbl: 192, 194
tc: 198, 199
tclsh: 121, 122
tclsh8.6: 121, 122
tee: 182, 186
telinit: 213, 217

telnet: 160, 161
test: 182, 186
texi2dvi: 206, 207
texi2pdf: 206, 207
texi2any: 206, 207
texindex: 206, 207
tfmtodit: 192, 194
tftp: 160, 161
tic: 145, 147
timedatectl: 213, 217
timeout: 182, 186
tload: 223, 224
toe: 145, 147
top: 223, 224
touch: 182, 186
tput: 145, 147
tr: 182, 186
traceroute: 160, 161
troff: 192, 194
true: 182, 186
truncate: 182, 186
tset: 145, 147
tsort: 182, 186
tty: 182, 186
tune2fs: 230, 232
tzselect: 101, 107
uclampset: 225, 229
udevadm: 213, 217
ul: 225, 229
umount: 225, 229
uname: 182, 186
uname26: 225, 229
uncompress: 197, 197
unexpand: 182, 186
unicode_start: 200, 201
unicode_stop: 200, 201
uniq: 182, 186
unlink: 182, 186
unlzma: 112, 113
unshare: 225, 229
unxz: 112, 113
updatedb: 190, 190
uptime: 223, 224
useradd: 135, 138
userdel: 135, 138
usermod: 135, 138
users: 182, 186
utmpdump: 225, 229
uuidd: 225, 229
uuidgen: 225, 229
uuidparse: 225, 229
vdir: 182, 186
vi: 208, 210
view: 208, 210
vigr: 135, 138
vim: 208, 210
vimdiff: 208, 210
vimtutor: 208, 210
vipw: 135, 138
vmstat: 223, 224
w: 223, 224
wall: 225, 229
watch: 223, 224
wc: 182, 186
wdctl: 225, 229
whatis: 220, 222
whereis: 225, 229
who: 182, 186
whoami: 182, 186
wipefs: 225, 229
x86_64: 225, 229
xargs: 190, 191
xgettext: 150, 151
xmlwf: 159, 159
xsubpp: 163, 165
xtrace: 101, 107
xxd: 208, 210
xz: 112, 113
xzcat: 112, 113
xzcmp: 112, 113
xzdec: 112, 113
xzdiff: 112, 113
xzegrep: 112, 113
xzfgrep: 112, 113
xzgrep: 112, 113
xzless: 112, 113
xzmore: 112, 113
yacc: 152, 152
yes: 182, 186
zcat: 197, 197
zcmp: 197, 197
zdiff: 197, 197
zdump: 101, 107
zegrep: 197, 197

zfgrep: 197, 197
zforce: 197, 197
zgrep: 197, 197
zic: 101, 107
zipdetails: 163, 165
zless: 197, 197
zmore: 197, 197
znew: 197, 197
zramctl: 225, 229
zstd: 114, 114
zstdgrep: 114, 114
zstdless: 114, 114

Libraries

Expat: 166, 166
ld-2.35.so: 101, 107
libacl: 133, 133
libanl: 101, 107
libasprintf: 150, 151
libattr: 132, 132
libbfd: 125, 127
libblkid: 225, 229
libBrokenLocale: 101, 107
libbz2: 110, 111
libc: 101, 107
libcap: 134, 134
libcheck: 187, 187
libcom_err: 230, 232
libcrypt: 101, 107
libcrypto.so: 171, 172
libctf: 125, 127
libctf-nobfd: 125, 127
libcursesw: 145, 147
libc_malloc_debug: 101, 107
libdbus-1: 218, 219
libdl: 101, 107
libe2p: 230, 232
libelf: 175, 175
libexpat: 159, 159
libexpect-5.45.4: 123, 123
libext2fs: 230, 232
libfdisk: 225, 229
libffi: 176
libfl: 120, 120
libformw: 145, 147
libg: 101, 107
libgcc: 139, 143
libgcov: 139, 143
libgdbm: 157, 157
libgdbm_compat: 157, 157
libgettextlib: 150, 151
libgettextpo: 150, 151
libgettextsrc: 150, 151
libgmp: 128, 129
libgmpxx: 128, 129
libgomp: 139, 143
libhistory: 116, 116
libitm: 139, 143
libkmod: 173
liblsan: 139, 143
libltdl: 156, 156
liblto_plugin: 139, 143
liblzma: 112, 113
libm: 101, 107
libmagic: 115, 115
libman: 220, 222
libmandb: 220, 222
libmcheck: 101, 107
libmemusage: 101, 107
libmenuw: 145, 147
libmount: 225, 229
libmpc: 131, 131
libmpfr: 130, 130
libmvec: 101, 107
libncursesw: 145, 147
libnsl: 101, 108
libnss_*: 101, 108
libopcodes: 125, 127
libpanelw: 145, 147
libpcprofile: 101, 108
libpipeline: 202
libprocps: 223, 224
libpsx: 134, 134
libpthread: 101, 108
libquadmath: 139, 143
libreadline: 116, 117
libresolv: 101, 108
librt: 101, 108
libsmartcols: 225, 229
libss: 230, 232
libssl.so: 171, 172
libssp: 139, 143
libstdbuf: 182, 186
libstdc++: 139, 143

libstdc++fs: 139, 143
libsubid: 135, 138
libsupc++: 139, 143
libsystemd: 213, 217
libtcl8.6.so: 121, 122
libtclstub8.6.a: 121, 122
libtextstyle: 150, 151
libthread_db: 101, 108
libtsan: 139, 143
libubsan: 139, 143
libudev: 213, 217
libutil: 101, 108
libuuid: 225, 229
liby: 152, 152
libz: 109, 109
libzstd: 114, 114
preloadable_libintl: 150, 151

Scripts

clock
  configuring: 244
console
  configuring: 245
hostname
  configuring: 239
localnet
  /etc/hosts: 239
network
  /etc/hosts: 239
  configuring: 236
network
  /etc/hosts: 239
  configuring: 236
dwp: 125, 127

Others

/boot/config-5.16.9: 256, 260
/boot/System.map-5.16.9: 256, 260
/dev/*: 75
/etc/fstab: 254
/etc/group: 78
/etc/hosts: 239
/etc/inputrc: 248
/etc/ld.so.conf: 106
/etc/lfs-release: 263
/etc/localtime: 104
/etc/lsb-release: 263
/etc/modprobe.d/usb.conf: 259
/etc/nsswitch.conf: 104
/etc/os-release: 263
/etc/passwd: 78
/etc/protocols: 100
/etc/resolv.conf: 238
/etc/services: 100
/etc/vimrc: 209
/run/utmp: 78
/usr/include/asm-generic/*.h: 49, 49
/usr/include/asm/*.h: 49, 49
/usr/include/drm/*.h: 49, 49
/usr/include/linux/*.h: 49, 49
/usr/include/misc/*.h: 49, 49
/usr/include/mtd/*.h: 49, 49
/usr/include/rdma/*.h: 49, 49
/usr/include/scsi/*.h: 49, 49
/usr/include/sound/*.h: 49, 49
/usr/include/video/*.h: 49, 49
/usr/include/xen/*.h: 49, 49
/var/log/btmp: 78
/var/log/lastlog: 78
/var/log/wtmp: 78
/etc/locale.conf: 246
/etc/shells: 250
man pages: 99, 99
Systemd Customization: 250