

Linux From Scratch

Versjon 12.3

Publisert 5. mars 2025

**Laget av Gerard Beekmans
Administrerende redaktør: Bruce Dubbs**

Linux From Scratch: Versjon 12.3: Publisert 5. mars 2025

av Laget av Gerard Beekmans og Administrerende redaktør: Bruce Dubbs
Opphavsrett © 1999-2025 Gerard Beekmans

Opphavsrett © 1999-2025, Gerard Beekmans

Alle rettigheter forbeholdt.

Denne boken er lisensiert under Creative Commons License.

Datainstruksjoner kan trekkes ut fra boken under MIT Lisensen.

Linux® er et registrert varemerke for Linus Torvalds.

Innholdsfortegnelse

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 verten	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 og Umask	18
2.7. Montering av den nye partisjonen	19
3. Pakker og oppdateringer	21
3.1. Introduksjon	21
3.2. Alle pakker	22
3.3. Nødvendige oppdateringer	30
4. Siste forberedelser	32
4.1. Introduksjon	32
4.2. Opprette et begrenset mappeoppsett i LFS filsystemet	32
4.3. Legge til LFS brukeren	32
4.4. Sette opp miljøet	33
4.5. Om SBU	35
4.6. Om testpakkene	36
III. Bygge LFS Kryssverktøykjede og midlertidige verktøy	38
Viktig foreløpig materiale	39
i. Introduksjon	39
ii. Verktøykjedens tekniske merknader	39
iii. Generelle kompilersinstruksjoner	44
5. Kompilere en kryssverktøykjede	46
5.1. Introduksjon	46
5.2. Binutils-2.44 - Pass 1	47
5.3. GCC-14.2.0 - Pass 1	49
5.4. Linux-6.13.4 API Deklarasjoner	52
5.5. Glibc-2.41	53

5.6. Libstdc++ fra GCC-14.2.0	56
6. Krysskompilering av midlertidige verktøy	57
6.1. Introduksjon	57
6.2. M4-1.4.19	58
6.3. Ncurses-6.5	59
6.4. Bash-5.2.37	61
6.5. Coreutils-9.6	62
6.6. Diffutils-3.11	63
6.7. File-5.46	64
6.8. Findutils-4.10.0	65
6.9. Gawk-5.3.1	66
6.10. Grep-3.11	67
6.11. Gzip-1.13	68
6.12. Make-4.4.1	69
6.13. Patch-2.7.6	70
6.14. Sed-4.9	71
6.15. Tar-1.35	72
6.16. Xz-5.6.4	73
6.17. Binutils-2.44 - Pass 2	74
6.18. GCC-14.2.0 - Pass 2	75
7. Gå inn i Chroot og bygge ytterligere midlertidige verktøy	77
7.1. Introduksjon	77
7.2. Skifte eierskap	77
7.3. Forberede det virtuelle kjernefilssystemet	77
7.4. Gå inn i Chroot miljøet	78
7.5. Opprette mapper	79
7.6. Opprette essensielle filer og symbolkoblinger	80
7.7. Gettext-0.24	83
7.8. Bison-3.8.2	84
7.9. Perl-5.40.1	85
7.10. Python-3.13.2	86
7.11. Texinfo-7.2	87
7.12. Util-linux-2.40.4	88
7.13. Rydde opp og lagre det midlertidige systemet	89
IV. Bygge LFS systemet	91
8. Installere grunnleggende systemprogramvare	92
8.1. Introduksjon	92
8.2. Pakkehåndtering	93
8.3. Man-pages-6.12	97
8.4. Iana-Etc-20250123	98
8.5. Glibc-2.41	99
8.6. Zlib-1.3.1	107
8.7. Bzip2-1.0.8	108
8.8. Xz-5.6.4	110
8.9. Lz4-1.10.0	112
8.10. Zstd-1.5.7	113
8.11. File-5.46	114
8.12. Readline-8.2.13	115
8.13. M4-1.4.19	117
8.14. Bc-7.0.3	118

8.15. Flex-2.6.4	119
8.16. Tcl-8.6.16	120
8.17. Expect-5.45.4	122
8.18. DejaGNU-1.6.3	124
8.19. Pkgconf-2.3.0	125
8.20. Binutils-2.44	126
8.21. GMP-6.3.0	129
8.22. MPFR-4.2.1	131
8.23. MPC-1.3.1	132
8.24. Attr-2.5.2	133
8.25. Acl-2.3.2	134
8.26. Libcap-2.73	135
8.27. Libxcrypt-4.4.38	136
8.28. Shadow-4.17.3	138
8.29. GCC-14.2.0	142
8.30. Ncurses-6.5	148
8.31. Sed-4.9	151
8.32. Psmisc-23.7	152
8.33. Gettext-0.24	153
8.34. Bison-3.8.2	155
8.35. Grep-3.11	156
8.36. Bash-5.2.37	157
8.37. Libtool-2.5.4	159
8.38. GDBM-1.24	160
8.39. Gperf-3.1	161
8.40. Expat-2.6.4	162
8.41. Inetutils-2.6	163
8.42. Less-668	165
8.43. Perl-5.40.1	166
8.44. XML::Parser-2.47	169
8.45. Intltool-0.51.0	170
8.46. Autoconf-2.72	171
8.47. Automake-1.17	172
8.48. OpenSSL-3.4.1	173
8.49. Libelf fra Elfutils-0.192	175
8.50. Libffi-3.4.7	176
8.51. Python-3.13.2	177
8.52. Flit-Core-3.11.0	179
8.53. Wheel-0.45.1	180
8.54. Setuptools-75.8.1	181
8.55. Ninja-1.12.1	182
8.56. Meson-1.7.0	183
8.57. Kmod-34	184
8.58. Coreutils-9.6	185
8.59. Check-0.15.2	190
8.60. Diffutils-3.11	191
8.61. Gawk-5.3.1	192
8.62. Findutils-4.10.0	193
8.63. Groff-1.23.0	194
8.64. GRUB-2.12	197

8.65. Gzip-1.13	199
8.66. IPRoute2-6.13.0	200
8.67. Kbd-2.7.1	202
8.68. Libpipeline-1.5.8	204
8.69. Make-4.4.1	205
8.70. Patch-2.7.6	206
8.71. Tar-1.35	207
8.72. Texinfo-7.2	208
8.73. Vim-9.1.1166	210
8.74. MarkupSafe-3.0.2	213
8.75. Jinja2-3.1.5	214
8.76. Udev fra Systemd-257.3	215
8.77. Man-DB-2.13.0	218
8.78. Procps-ng-4.0.5	221
8.79. Util-linux-2.40.4	223
8.80. E2fsprogs-1.47.2	228
8.81. Sysklogd-2.7.0	231
8.82. SysVinit-3.14	232
8.83. Om feilsøkingssymboler	233
8.84. Stripping	233
8.85. Rydde opp	235
9. Systemkonfigurasjon	236
9.1. Introduksjon	236
9.2. LFS-Bootscripts-20240825	237
9.3. Oversikt over enhets- og modulhåndtering	239
9.4. Administrere enheter	241
9.5. Generell nettverkskonfigurasjon	244
9.6. System V Oppstartskript Bruk og Konfigurasjon	247
9.7. Konfigurere systemlokaliteten	255
9.8. Opprette /etc/inputrc filen	256
9.9. Opprette /etc/shells filen	257
10. Gjøre LFS systemet oppstartbart	259
10.1. Introduksjon	259
10.2. Opprette /etc/fstab filen	259
10.3. Linux-6.13.4	261
10.4. Bruke GRUB til å sette opp oppstartsprosessen	267
11. Slutt	270
11.1. Slutt	270
11.2. Bli regnet med	270
11.3. Omstart av systemet	270
11.4. Tilleggsressurser	271
11.5. Komme i gang etter LFS	272
V. Vedlegg	275
A. Akronymer og begreper	276
B. Anerkjennelser	278
C. Avhengigheter	281
D. Oppstarts og sysconfig skriptversjon-20240825	295
D.1. /etc/rc.d/init.d/rc	295
D.2. /lib/lsb/init-functions	298
D.3. /etc/rc.d/init.d/mountvirtfs	311

D.4. /etc/rc.d/init.d/modules	313
D.5. /etc/rc.d/init.d/udev	314
D.6. /etc/rc.d/init.d/swap	315
D.7. /etc/rc.d/init.d/setclock	316
D.8. /etc/rc.d/init.d/checkfs	317
D.9. /etc/rc.d/init.d/mountfs	319
D.10. /etc/rc.d/init.d/udev_retry	321
D.11. /etc/rc.d/init.d/cleanfs	322
D.12. /etc/rc.d/init.d/console	324
D.13. /etc/rc.d/init.d/localnet	325
D.14. /etc/rc.d/init.d/sysctl	326
D.15. /etc/rc.d/init.d/syslogd	327
D.16. /etc/rc.d/init.d/network	328
D.17. /etc/rc.d/init.d/sendsignals	330
D.18. /etc/rc.d/init.d/reboot	331
D.19. /etc/rc.d/init.d/halt	332
D.20. /etc/rc.d/init.d/template	332
D.21. /etc/sysconfig/modules	333
D.22. /etc/sysconfig/createfiles	334
D.23. /etc/sysconfig/udev-retry	334
D.24. /sbin/ifup	335
D.25. /sbin/ifdown	337
D.26. /lib/services/ipv4-static	338
D.27. /lib/services/ipv4-static-route	340
E. Udev konfigurasjonsregler	342
E.1. 55-lfs.rules	342
F. LFS lisenser	343
F.1. Creative Commons License	343
F.2. The MIT License	347
Register	348

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 alle 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 ha 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 spør 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. *Du* dikterer alle aspekter av systemet ditt.

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 størrelshensyn 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 problemet 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

Den primære målarkiturene 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 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.

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 minutter	3.6 GB
64-bit	233.2 minutter	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, en gang for et 32-bitssystem og en 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 riktig utføre kommandoene som er oppført. Spesielt som et absolutt minimum, bør du allerede vite hvordan du bruker kommandolinjen (skallet) for å kopiere eller flytte filer og mapper, liste mapper og filinnhold, og endre gjeldende mappe. 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 sannsynligvis vil 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 <https://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 <https://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) Versjon 3.0*
- *Linux Standard base (LSB) Versjon 5.0 (2015)*

LSB har fire separate standarder: Kjerne, Skrivebord, Språk 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 samsvare med LSB spesifikasjoner for IA32 (32-bit x86) eller AMD64 (x86_64) arkitekturer omtalt i forrige avsnitt.



Notat

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. Installasjonsveiledning for noen av disse tilleggspakker finnes i BLFS.

Pakker levert av LFS som trengs for å tilfredsstillere LSB kravene

LSB Kjerne:

Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Gzip, M4, Man-DB, Procps, Psmisc, Sed, Shadow, SysVinit, Tar, Util-linux, Zlib

LSB Skrivebord:

Ingen

<i>LSB Språk:</i>	Perl
<i>LSB Bildebehandling:</i>	Ingen
<i>LSB Gtk3 og LSB Grafikk (Prøvebruk):</i>	Ingen

Pakker levert av BLFS som trengs for å tilfredsstille LSB kravene

<i>LSB Kjerne:</i>	At, Batch (en del av At), BLFS Bash Startup Files, Cpio, Ed, Fcfrontab, LSB-Tools, NSPR, NSS, Linux-PAM, Pax, Sendmail (eller Postfix eller Exim), Time
<i>LSB Skrivebord:</i>	Alsa, ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Gdk-pixbuf, Glib2, GLU, Icon-naming-utils, Libjpeg-turbo, Libxml2, Mesa, Pango, Xdg-utils, Xorg
<i>LSB Språk:</i>	Libxml2, Libxslt
<i>LSB Bildebehandling:</i>	CUPS, Cups-filters, Ghostscript, SANE
<i>LSB Gtk3 and LSB Grafikk (Prøvebruk):</i>	GTK+3

Komponenter som ikke følger med eller valgfritt levert av LFS eller BLFS som trengs for å tilfredsstille LSB Krav

<i>LSB Kjerne:</i>	install_initd , libcrypt.so.1 (kan leveres med valgfrie instruksjoner for LFS Libxcrypt pakken), libncurses.so.5 (kan leveres med valgfrie instruksjoner for LFS Ncurses pakken), libncursesw.so.5 (men libncursesw.so.6 leveres av LFS Ncurses pakken)
<i>LSB Skrivebord:</i>	libgdk-x11-2.0.so (men libgdk-3.so leveres av BLFS GTK+-3 pakken), libgtk-x11-2.0.so (men libgtk-3.so og libgtk-4.so leveres av BLFS GTK+-3 and GTK-4 pakkene), libpng12.so (men libpng16.so leveres av BLFS Libpng pakken), libQt*.so.4 (men libQt6*.so.6 leveres av BLFS Qt6 pakken), libtiff.so.4 (men libtiff.so.6 leveres av BLFS Libtiff pakken)
<i>LSB Språk:</i>	/usr/bin/python (LSB krever Python2 men LFS og BLFS leverer bare Python3)
<i>LSB Bildebehandling:</i>	Ingen
<i>LSB Gtk3 and LSB Grafikk (Prøvebruk):</i>	libpng15.so (men libpng16.so leveres av BLFS Libpng pakken)

Begrunnelse for pakker i boken

Målet med LFS er å bygge en komplett og brukbart system på fundamentnivå—inkludert alle pakkene som trengs for å replikere seg selv—og gi en relativt minimal base for å tilpasse et mer komplett system basert på brukerens valg. Dette betyr ikke at LFS er det minste systemet som er mulig. Flere viktige pakker er inkludert som strengt tatt ikke er påkrevd. Listen nedenfor dokumenterer grunner til at hver pakke i boken er inkludert.

- Acl

Denne pakken inneholder verktøy for å administrere tilgangskontrollister, som brukes til å definere mer finkornet skjønsmessige 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 kildekode 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 tilfredsstillter 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 tilfredsstillter 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.

- 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.

- 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 (lexical analyzer) 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 bygges av LFS.

- GDBM

Denne pakken inneholder GNU biblioteket for databasebehandling. Den brukes av en annen LFS pakke, MandB.

- 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 hashfunksjon fra et nøkkelsett. Den kreves av Udev .

- Grep

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

- Groff

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

- 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.

- 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 den andre vanlige verktøypakken for nettverk (net-tools) for sine IPv6-funksjoner.

- Kbd

Denne pakken inneholder tastaturlistefiler, 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. Mange pakker bruker den til å søke på utdataene.

- 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 compilert kode.

- Libpipeline

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

- Libtool

Denne pakken inneholder GNU skriptet for generisk bibliotekstøtte. Det omslutter kompleksiteten ved å bruke delte biblioteker i en konsekvent, flyttbart grensesnitt. Det trengs av testpakker i andre LFS pakker.

- Libxcrypt

Denne pakken gir `libcrypt` bibliotek nødvendig av forskjellige pakker (spesielt Shadow) for hashing av passord. Den erstatter det foreldede `libcrypt` implementering i Glibc.

- Linux Kernel

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

- M4

Denne pakken inneholder en generell tekstmakroprossessor 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.

- Man-DB

Denne pakken inneholder programmer for å finne og vise mansider. 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 grunnleggende mansider for Linux.

- 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 byggesystemet. 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 multipresisjons 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 byggesystem, 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 Linuxkjernen.

- Patch

Denne pakken inneholder et program for å endre eller lage filer ved å bruke en *oppdateringsfil* (*patch*) 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.

- Pkgconf

Denne pakken inneholder et program som hjelper til med å konfigurere kompilator- og linkerflagg for utviklingsbiblioteker. Programmet kan brukes som drop-in erstatning for **pkg-config**, som trengs av byggesystemet for mange pakker. Det vedlikeholdes mer aktivt og er litt raskere enn den originale Pkg-config pakken.

- 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 systemadministrasjon.

- 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 konfigureringskript.

- Shadow

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

- Sysklogd

Denne pakken inneholder programmer for logging av systemmeldinger, slik som de som er gitt av kjernen eller nisseprosessene (daemon processes) når uvanlig hendelser oppstår.

- SysVinit

Denne pakken gir init programmet, som er overordnet for alle andre prosesser på et Linux system.

- Udev

Denne pakken er en enhetsbehandler. Den styrer dynamisk eierskapet, tillatelser, navn og symbolske lenker til enheter i /dev mappen mens enheter legges til eller fjernes fra systemet.

- Tar

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

- Tcl

Denne pakken inneholder Verktøykommandospråk (Tool Command Language) som brukes i mange testpakker i LFS pakker.

- Texinfo

Denne pakken inneholder programmer for å lese, skrive til og konvertere informasjonssider. Den brukes i installasjonsprosedyrer 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 enormt antall kraftige kapasiteter. Et redigeringsprogram er et veldig personlig valg for mange brukere og andre redigeringsprogram kan brukes om ønskelig.

- Wheel

Denne pakken inneholder Python modulen Wheel som er referanseimplementering av Python wheel pakkingsstandarden.

- 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 et svært bredt utvalg av kompresjon/hastighets avveininger.

Typografi

For å gjøre ting lettere å følge, er 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. Den 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 tabulator tegn vil lage feil resultater.

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

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



Notat

Vennligst konfigurer nettleseren din til å vise tekst med fast bredde og en god monospace" font-size="9pt font, så du kan skille tegnvariantene `111` eller `00` helt klart.

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.

<ERSTATTET TEKST>

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 (manside). Tallet innenfor parentes indikerer en bestemt del i håndbøkene. For eksempel, **passwd** har to mandsider. I henhold til LFS installasjonsinstruksjoner, disse to mandsidene 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 til `/usr/share/man/man5/passwd.5`. **man passwd** vil skrive ut den første mandsiden den finner som stemmer med «passwd», som vil bli `/usr/share/man/man1/passwd.1`. For dette eksemplet trenger du å kjøre **man 5 passwd** for å lese siden som blir spesifisert. Merk at de fleste mandsider ikke har duplikate sidenavn i forskjellige seksjoner. Derfor, **man** <programnavn> er generelt tilstrekkelig. I LFS boken er disse referansene til mandsider også hyperkoblinger, så klikk på en slik referanse vil åpne mandsiden gjengitt i HTML fra *Arch Linux manual pages*.

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 en etter en, 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 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 boken, 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—å reparere sikkerhetssårbarheter eller for å fikse andre feil—besøk <https://www.linuxfromscratch.org/lfs/errata/12.3/> før du fortsetter med byggingen. Du bør merke deg endringer som vises, og bruke dem på den relevante delen av boken mens du bygger LFS systemet.

I tillegg opprettholder Linux From Scratch redaktørene en liste over sikkerhetssårbarheter oppdaget *etter* at en bok ble utgitt. For å lese listen, besøk <https://www.linuxfromscratch.org/lfs/advisories/> før du fortsetter med byggingen. Du bør anvende endringene foreslått av rådene til de relevante delene av boken mens du bygger LFS systemet. Og hvis du vil bruke LFS systemet som et ekte skrivebord eller serversystem, bør du fortsette å konsultere råd og fikse eventuelle sikkerhetssårbarheter, selv når LFS systemet er ferdig bygget.

Del I. Introduksjon

Kapittel 1. Introduksjon

1.1. Hvordan bygge et LFS-system

LFS systemet vil bli bygget ved å bruke en allerede installert Linuxdistribusjon (som Debian, OpenMandriva, Fedora eller openSUSE). Denne eksisterende Linuxsystemet (verten) vil bli brukt som utgangspunkt for å gi nødvendige programmer, inkludert en kompilator, linker og skall, for å bygge det nye systemet. Velg «development» alternativet under distribusjonsinstallasjonen for å kunne få tilgang til disse verktøyene.



Notat

Det er mange måter å installere en Linuxdistribusjon på og standardinnstillingene er vanligvis ikke optimale for å bygge et LFS system. For forslag til å sette opp en kommersiell distribusjon, se: <https://www.linuxfromscratch.org/hints/downloads/files/partitioning-for-lfs.txt>.

Som et alternativ til å installere en separat distribusjon på din maskin, kan det være lurt 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 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 krysskompileringsteknikker 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 et "chroot" miljø, der vi bruker de nye verktøyene til å bygge resten av verktøyene som trengs for å lage LFS systemet.

Denne innsatsen for å isolere det nye systemet fra vertsdistribusjonen kan virke overdreven. En fullstendig teknisk forklaring på hvorfor dette gjøres er gitt i Verktøykjedens tekniske merknader.

I Kapittel 8 blir det fulle LFS system 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 blir opprettet 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

Her er en liste over pakkene som er oppdatert siden forrige utgivelse av LFS.

Oppgradert til:

-
- Bash-5.2.37

- Bc-7.0.3
- Binutils-2.44
- Coreutils-9.6
- Diffutils-3.11
- E2fsprogs-1.47.2
- Expat-2.6.4
- File-5.46
- Flit-core-3.11.0
- Gawk-5.3.1
- Gettext-0.24
- Glibc-2.41
- Iana-Etc-20250123
- Inetutils-2.6
- IPRoute2-6.13.0
- Jinja2-3.1.5
- Kbd-2.7.1
- Kmod-34
- Less-668
- Libcap-2.73
- Libelf from Elfutils-0.192
- Libffi-3.4.7
- Libpipeline-1.5.8
- Libtool-2.5.4
- Libxcrypt-4.4.38
- Linux-6.13.4
- Man-DB-2.13.0
- Man-pages-6.12
- MarkupSafe-3.0.2
- Meson-1.7.0
- OpenSSL-3.4.1
- Perl-5.40.1
- Procps-ng-4.0.5
- Python-3.13.2
- Setuptools-75.8.1
- Shadow-4.17.3
- Sysklogd-2.7.0
- Systemd-257.3
- SysVinit-3.14
- Tcl-8.6.16
- Texinfo-7.2
- Tzdata-2025a
- Udev from Systemd-257.3
- Util-linux-2.40.4

- Vim-9.1.1166
- Wheel-0.45.1
- Xz-5.6.4
- Zstd-1.5.7

Lagt til:

-

Fjernet:

-

1.3. Endringslogg

Dette er versjon 12.3 av Linux From Scratch-boken, datert 5. mars 2025. Hvis denne boken er mer enn seks måneder gammel, er en nyere og bedre versjonen 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:

- 05.03.2025
 - [bdubbs] - LFS-12.3 utgitt.
- 02.03.2025
 - [bdubbs] - Oppdater til vim-9.1.1166 (Sikkerhetsoppdatering). Fikser #5666.
- 27.02.2025
 - [bdubbs] - Oppdater til zstd-1.5.7. Fikser #5652.
 - [bdubbs] - Oppdater til systemd-257.3. Fikser #5612.
 - [bdubbs] - Oppdater til shadow-4.17.3. Fikser #5660.
 - [bdubbs] - Oppdater til setup-tools-75.8.1. Fikser #5662.
 - [bdubbs] - Oppdater til linux-6.13.4. Fikser #5647.
 - [bdubbs] - Oppdater til kmod-34. Fikser #5657.
 - [bdubbs] - Oppdater til inetutils-2.6. Fikser #5656.
 - [bdubbs] - Oppdater til gettext-0.24. Fikser #5661.
 - [bdubbs] - Oppdater til flit_core-3.11.0. Fikser #5654.
- 24.02.2025
 - [bdubbs] - Oppdater til man-pages-6.12. Fikser #5658.
- 19.02.2025
 - [xry111] - Oppdater til vim-9.1.1122 (Sikkerhetsoppdatering). Adresserer #4500.
 - [xry111] - Oppdater til man-pages-6.11. Fikser #5646.
- 13.02.2025
 - [bdubbs] - Oppdater til vim-9.1.1106. Adresserer #4500.
 - [bdubbs] - Oppdater til diffutils-3.11. Fikser #5639.
 - [bdubbs] - Oppdater til libffi-3.4.7. Fikser #5642.
 - [bdubbs] - Oppdater til linux-6.13.2. Fikser #5643.
 - [bdubbs] - Oppdater til Python3-3.13.2. Fikser #5640.

- [bdubbs] - Oppdater til sysvinit-3.14. Fikser #5641.
- 02.02.2025
 - [bdubbs] - Oppdater til vim-9.1.1071. Adresserer #4500.
 - [bdubbs] - Oppdater til iana-etc-20250123. Adresserer #5006.
 - [bdubbs] - Oppdater til binutils-2.44.0. Fikser #5634.
 - [bdubbs] - Oppdater til coreutils-9.6. Fikser #5628.
 - [bdubbs] - Oppdater til e2fsprogs-1.47.2. Fikser #5637.
 - [bdubbs] - Oppdater til glibc-2.41. Fikser #5638.
 - [bdubbs] - Oppdater til iproute2-6.13.0. Fikser #5631.
 - [bdubbs] - Oppdater til libxcrypt-4.4.38. Fikser #5626.
 - [bdubbs] - Oppdater til linux-6.13.1. Fikser #5629.
 - [bdubbs] - Oppdater til man-pages-6.10. Fikser #5632.
 - [bdubbs] - Oppdater til meson-1.7.0. Fikser #5636.
 - [bdubbs] - Oppdater til perl-5.40.1. Fikser #5630.
 - [bdubbs] - Oppdater til tcl8.6.16. Fikser #5635.
 - [bdubbs] - Oppdater til tzdata2025a. Fikser #5627.
 - [bdubbs] - Oppdater til xz-5.6.4. Fikser #5633.
- 15.01.2025
 - [bdubbs] - Oppdater til vim-9.1.1016. Adresserer #4500.
 - [bdubbs] - Oppdater til iana-etc-20250108. Adresserer #5006.
 - [bdubbs] - Oppdater til util-linux-2.40.4. Fikser #5624.
 - [bdubbs] - Oppdater til sysvinit-3.13. Fikser #5621.
 - [bdubbs] - Oppdater til sysklogd-2.7.0. Fikser #5623.
 - [bdubbs] - Oppdater til shadow-4.17.2. Fikser #5625.
 - [bdubbs] - Oppdater til setuptools-75.8.0. Fikser #5622.
 - [bdubbs] - Oppdater til linux-6.12.9. Fikser #5620.
 - [bdubbs] - Oppdater til gettext-0.23.1. Fikser #5619.
- 01.01.2025
 - [renodr] - Oppdater til libxcrypt-4.4.37. Fikser #5618.
 - [bdubbs] - Oppdater til iana-etc-20241220. Adresserer #5006.
 - [bdubbs] - Oppdater til texinfo-7.2. Fikser #5616.
 - [bdubbs] - Oppdater til sysvinit-3.12. Fikser #5615.
 - [bdubbs] - Oppdater til shadow-4.17.1. Fikser #5617.
 - [bdubbs] - Oppdater til procps-ng-4.0.5. Fikser #5611.
 - [bdubbs] - Oppdater til meson-1.6.1. Fikser #5610.
 - [bdubbs] - Oppdater til linux-6.12.7. Fikser #5613.
 - [bdubbs] - Oppdater til kbd-2.7.1. Fikser #5608. 5
 - [bdubbs] - Oppdater til jinja2-3.1.5 (Sikkerhetsoppdatering). Fikser #5614.

- 15.12.2024
 - [bdubbs] - Oppdater til vim-9.1.0927. Adresserer #4500.
 - [bdubbs] - Oppdater til iana-etc-20241206. Adresserer #5006.
 - [bdubbs] - Oppdater til systemd-257. Fikser #5559.
 - [bdubbs] - Oppdater til Python-3.13.1 (Sikkerhetsoppdatering). Fikser #5605.
 - [bdubbs] - Oppdater til libcap-2.73. Fikser #5604.
 - [bdubbs] - Oppdater til linux-6.12.5. Fikser #5607.
 - [bdubbs] - Oppdater til kbd-2.7. Fikser #5608.
 - [bdubbs] - Oppdater til gettext-0.23. Fikser #5603.
- 01.12.2024
 - [bdubbs] - Oppdater til iana-etc-20241122. Adresserer #5006.
 - [bdubbs] - Oppdater til file-5.46. Fikser #5601.
 - [bdubbs] - Oppdater til iproute2-6.12.0. Fikser #5597.
 - [bdubbs] - Oppdater til libtool-2.5.4. Fikser #5598.
 - [bdubbs] - Oppdater til linux-6.12.1. Fikser #5586.
 - [bdubbs] - Oppdater til setuptools-75.6.0 (Python Modul). Fikser #5599.
 - [bdubbs] - Oppdater til wheel-0.45.1 (Python Modul). Fikser #5600.
- 15.11.2024
 - [bdubbs] - Oppdater til vim-9.1.0866. Adresserer #4500.
 - [bdubbs] - Oppdater til iana-etc-20241024. Adresserer #5006.
 - [bdubbs] - Oppdater til wheel-0.45.0 (Python Modul). Fikser #5593.
 - [bdubbs] - Oppdater til setuptools-75.5.0 (Python Modul). Fikser #5595.
 - [bdubbs] - Oppdater til linux-6.11.8. Fikser #5582.
 - [bdubbs] - Oppdater til libcap-2.72. Fikser #5594.
- 08.11.2024
 - [bdubbs] - Legg til binutils-2.43.1-upstream_fix-1.patch. Fikser #5591.
 - [bdubbs] - Oppdater til flit_core-3.10.1. Fikser #5589.
 - [bdubbs] - Oppdater til expat-2.6.4. Fikser #5590.
- 25.10.2024
 - [bdubbs] - Oppdater til linux-6.11.6. Fikser #5588.
 - [bdubbs] - Oppdater til libcap-2.71. Fikser #5584.
 - [bdubbs] - Oppdater til setuptools-75.3.0. Fikser #5585.
 - [bdubbs] - Oppdater til flit_core-3.10.0. Fikser #5587.
- 25.10.2024
 - [bdubbs] - Oppdater til iana-etc-20241015. Adresserer #5006.
 - [bdubbs] - Oppdater til vim-9.1.0813. Adresserer #4500.
 - [bdubbs] - Oppdater til xz-5.6.3. Fikser #5572.
 - [bdubbs] - Oppdater til sysvinit-3.11. Fikser #5581.

- [bdubbs] - Oppdater til setuptools-75.2.0. Fikser #5577.
- [bdubbs] - Oppdater til Python3-3.13.0. Fikser #5575.
- [bdubbs] - Oppdater til openssl-3.4.0. Fikser #5582.
- [bdubbs] - Oppdater til meson-1.6.0. Fikser #5580.
- [bdubbs] - Oppdater til markupsafe-3.0.2. Fikser #5576.
- [bdubbs] - Oppdater til linux-6.11.5. Fikser #5574.
- [bdubbs] - Oppdater til less-668. Fikser #5578.
- [bdubbs] - Oppdater til elfutils-0.192. Fikser #5579.
- 03.10.2024
 - [bdubbs] - Gå tilbake til tcl8.6.15.
- 01.10.2024
 - [bdubbs] - Oppdater til Python3-3.12.7. Fikser #5571.
 - [bdubbs] - Oppdater til tcl9.0.0. Fikser #5570.
 - [bdubbs] - Oppdater til linux-6.11.1. Fikser #5556.
 - [bdubbs] - Oppdater til libtool-2.5.3. Fikser #5569.
 - [bdubbs] - Oppdater til iproute2-6.11.0. Fikser #5561.
 - [bdubbs] - Oppdater til bash-5.2.37. Fikser #5567.
 - [bdubbs] - Oppdater til bc-7.0.3. Fikser #5568.
- 20.09.2024
 - [bdubbs] - Oppdater til vim-9.1.0738. Adresserer #4500.
 - [bdubbs] - Oppdater til texinfo-7.1.1. Fikser #5558.
 - [bdubbs] - Oppdater til tcl8.6.15. Fikser #5562.
 - [bdubbs] - Oppdater til sysklogd-2.6.2. Fikser #5557.
 - [bdubbs] - Oppdater til setuptools-75.1.0. Fikser #5560.
 - [bdubbs] - Oppdater til meson-1.5.2. Fikser #5566.
 - [bdubbs] - Oppdater til iana-etc-20240912. Adresserer #5006.
 - [bdubbs] - Oppdater til gawk-5.3.1. Fikser #5564.
 - [bdubbs] - Oppdater til bc-7.0.2. Fikser #5563.
- 07.09.2024
 - [bdubbs] - Oppdater til tzdata-2024b. Fikser #5554.
 - [bdubbs] - Oppdater til systemd-256.5. Fikser #5551.
 - [bdubbs] - Oppdater til setuptools-74.1.2. Fikser #5546.
 - [bdubbs] - Oppdater til python3-3.12.6. Fikser #5555.
 - [bdubbs] - Oppdater til openssl-3.3.2. Fikser #5552.
 - [bdubbs] - Oppdater til man-db-2.13.0. Fikser #5550.
 - [bdubbs] - Oppdater til linux-6.10.8. Fikser #5545.
 - [bdubbs] - Oppdater til libpipeline-1.5.8. Fikser #5548.
 - [bdubbs] - Oppdater til expat-2.6.3. Fikser #5553.7

- [bdubbs] - Oppdater til bc-7.0.1. Fikser #5547.
- 01.09.2024
- [bdubbs] - LFS-12.2 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, arkivsteder 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. 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



Notat

I tilfelle du har et problem med å bygge en pakke med LFS instruksjoner fraråder vi på det sterkeste å legge ut problemet direkte på oppstrøms støttekanalen før diskusjon via en LFS støttekanal oppført i Seksjon 1.4, «Ressurser». Å gjøre det er ofte ganske ineffektivt fordi oppstrøms vedlikeholdere sjelden er kjent med LFS byggeprosedyre. Selv om du virkelig har truffet et oppstrømsproblem, kan LFS-fellesskapet fortsatt hjelpe å isolere informasjonen ønsket av oppstrøms vedlikeholdere og lage en skikkelig rapport.

Hvis du må stille et spørsmål direkte via en oppstrøms støttekanal, skal du i det minste merke deg at mange oppstrømsprosjekter har støttekanaler atskilt fra feilsporeren. «bug» rapporter for å stille spørsmål anses som ugyldige og kan irritere oppstrømsutviklere for disse prosjektene.

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 FAQ, 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 Seksjon 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, enhver forespørsel om hjelp bør inkludere disse viktige tingene:

- Versjonen av boken som brukes (i dette tilfellet 12.3)
- Vertsdistribusjonen og versjonen som brukes til å lage LFS
- Utdata fra Systemkrav for verten skriptet
- Pakken eller seksjonen problemet ble oppdaget i
- Den nøyaktige feilmeldingen, eller en tydelig beskrivelse av problemet
- Gi beskjed om du i det hele tatt har avveket fra boken



Notat

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** skriptet, 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 -D ALIASEPATH="/mnt/lfs/usr/share/locale:."
-D LOCALEDIR="/mnt/lfs/usr/share/locale"
-D LIBDIR="/mnt/lfs/usr/lib"
-D INCLUDEDIR="/mnt/lfs/usr/include" -D HAVE_CONFIG_H -I. -I.
-g -O2 -c getopt1.c
gcc -g -O2 -static -o make ar.o arscan.o commands.o dir.o
expand.o file.o function.o getopt.o implicit.o job.o main.o
misc.o read.o remake.o rule.o 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.

Del II. Forbereder for byggingen

Kapittel 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, lager et filsystem på den og monter den.

2.2. Systemkrav for verten

2.2.1. Maskinvare

LFS redaktørene anbefaler at CPUen har minst fire kjerner og at systemet har minst 8 GB minne. Eldre systemer som ikke oppfyller disse kravene vil fortsatt fungere, men tiden for å bygge pakker vil bli betydelig lengre enn dokumentert.

2.2.2. Programvare

Vertssystemet ditt bør ha følgende programvare med minimumsversjoner angitt. Dette burde ikke være et problem for de fleste moderne Linuxdistribusjoner. Vær også oppmerksom på at mange distribusjoner vil plassere programvaredeklarasjoner i separate pakker, ofte i form av `<pakkenavn>-devel` eller `<pakkenavn>-dev`. Sørg for 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 2.44 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-8.1**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk bør være en link til gawk)
- **GCC-5.2** inkludert C++ kompilatoren, `g++` (Versjoner større enn 14.2.0 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-5.4**

Grunnen til kravet om kjerneversjon er at vi spesifiserer den versjonen når du bygger glibc i Kapittel 5 og Kapittel 8, så løsningene for eldre kjerner er ikke aktivert og det kompilerte glibc er litt raskere og mindre. Per desember 2024, 5.4 er fortsatt den eldste kjerneutgivelsen støttet av kjerneutviklerne. Noen kjerneutgivelser som er eldre enn 5.4 kan fortsatt støttes av tredjepartsteam, men de regnes ikke som offisielle oppstrøms kjerneutgivelser; les <https://kernel.org/category/releases.html> for detaljer.

Hvis vertskjernen er tidligere enn 5.4 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 5.4 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 installere den, kan du compilere en kjerne selv. Instruksjoner for å compilere kjernen og konfigurere oppstartslasteren (forutsatt at verten bruker GRUB) er lokalisert i Kapittel 10.

Vi krever at vertskjernen støtter UNIX 98 pseudoterminal (PTY). Det bør være aktivert på all distribusjoner, skrivebord eller server med Linux 5.4 eller en nyere kjerne. Hvis du bygger en egendefinert vertskjerne, sørg for at `CONFIG_UNIX98_PTYS` er satt til `y` i kjernekonfigurasjonen.

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



Viktig

Merk at symbolenkene nevnt ovenfor er nødvendige for å bygge et LFS system ved å bruke instruksjonene i denne boken. Sybollinker 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
# A script to list version numbers of critical development tools

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

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

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

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

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



```

}

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

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

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

echo "Compiler check:"
if printf "int main(){}" | g++ -x c++ -
then echo "OK: g++ works";
else echo "ERROR: g++ does NOT work"; fi
rm -f a.out
if [ "$(nproc)" = "" ]; then
    echo "ERROR: nproc is not available or it produces empty output"
else
    echo "OK: nproc reports $(nproc) logical cores are available"
fi
EOF

bash version-check.sh

```

2.3. Bygge LFS i etapper

LFS er designet for å bygges i en økt. Det er det instruksjonene forutsetter, at systemet ikke vil bli slått av under prosessen. Det betyr ikke at byggingen av 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 sikker på en ting:

- 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** kommando 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 Generelle kompilersinstruksjoner er kritiske. Hvis det er noen tvil om installerte pakker, sørg for at tidligere utpakkede tarballer 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 «Forberede det virtuelle kjernefilssystemet» for å «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 for `root`. LFS variabelen brukes ikke etter at du er gått inn i `chroot` miljøet.
- 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`, kjører kommandoene i Seksjon 7.3.1, «Montering og fylling av /dev» og Seksjon 7.3.2, «Montering av det virtuelle kjernefilssystemer»

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 kompilersprosesser er det en god ide å bruke en liten diskpartisjon som `swap`. 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 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 ennå ikke vet hvordan du bruker programmene.



Notat

For erfarne brukere er andre partisioneringsordninger 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 partisioneringsmetodene 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 root 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 diskveksling.

Hvis du vil bruke dvalefunksjonen (suspend-to-disk) i Linux, den skriver ut innholdet i RAM til vekselminnepartisjonen får 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. Med en SSD vil du ikke kunne høre veksling, men du kan se hvor mye vekslingsplass som brukes ved å kjøre **top** eller **free** programmene. Bruken av en SSD 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 er normalt, er den beste løsningen å kjøpe mer RAM til ditt system.

2.4.1.3. Grub Bios partisjonen

Hvis *boot disk* 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 oppstartslasteren. Denne partisjonen vil normalt være merket 'BIOS Boot' hvis den opprettes av **fdisk** eller har en kode på *EF02* hvis du bruker **gdisk** kommandoen.



Notat

Grub Bios partisjonen må være på stasjonen som BIOS bruker for å starte opp systemet. Dette er ikke nødvendigvis den stasjonen som holder LFS rotpartisjon. Disker på et system kan bruke forskjellig partisjonstabelltyper. Nødvendigheten av Grub Bios partisjonen avhenger bare på partisjonstabelltypen til oppstartsdisk.

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 oppstartsproblemer med større disk, gjør dette til den første fysiske partisjonen på din første diskstasjon. En partisjonsstørrelse på 200 megabyte er tilstrekkelig.
- `/boot/efi` – EFI systempartisjonen, som er nødvendig for å starte opp systemet med UEFI. Les *BLFS siden* for detaljer.

- /home – Sterkt anbefalt. Del hjemmemappen 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 motparter 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 lager det uansett, 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 en gigabyte) i denne konfigurasjonen, så det er egnet for en tynnklient eller diskløs arbeidsstasjon (hvor /usr monteres fra en fjernserver). Du bør imidlertid være oppmerksom på at initramfs (ikke dekket av LFS) vil være nødvendig for å starte et system med en separat /usr partisjon.
- /opt – Denne mappen er mest nyttig for BLFS der flere store pakker som KDE eller Texlive kan installeres uten å bygge inn filene i /usr hierarkiet. Hvis den brukes, er 5 til 10 gigabyte generelt tilstrekkelig.
- /tmp – En separat /tmp partisjon er sjeldent, men nyttig hvis du konfigurerer en tynnklient. Denne partisjonen, hvis brukt, vil vanligvis ikke trenge å overstige et par gigabyte. Hvis du har nok RAM, kan du montere en `tmpfs` på /tmp å lage tilgangen til midlertidige filer raskere.
- /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 som du ønsker automatisk montert når systemet starter må spesifiseres i `/etc/fstab` file. Detaljer om hvordan du spesifiserer partisjoner vil bli diskutert i Seksjon 10.2, «Opprette `/etc/fstab` file».

2.5. Opprette et filsystem på partisjonen

En partisjon er bare en rekke sektorer på en diskstasjon, avgrenset med grenser satt i en partisjonstabell. Før operativsystemet kan bruke en partisjon for å lagre alle filer, må partisjonen formateres til å inneholde et filsystem, vanligvis bestående av en etikett, katalogblokker, datablokker og et indekseringsskjema for å finne en bestemt fil på forespørsel. Filsystemet hjelper også OS med å holde styr på ledig plass på partisjonen, reservere nødvendige sektorer når en ny fil opprettes eller en eksisterende fil utvides, og resirkuler de ledige datasegmentene som opprettes når filer slettes. Det kan også gi støtte for dataredundans og for feilgjenoppretting.

LFS kan bruke et hvilket som helst filsystem som gjenkjennes av Linuxkjernen, men de vanligste typene er `ext3` og `ext4`. Valget av riktig filsystem kan være kompleks; det 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`-familien til filsystemer. Det gir flere nye funksjoner, inkludert nano-sekunders tidsstempler, opprettelse og bruk av svært store filer (opptil 16 TB), og hastighetsforbedringer.

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

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

```
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 og Umask

Gjennom hele denne boken, vil miljøvariabelen `LFS` brukes flere ganger. Du bør sørge for at denne variabelen alltid er definert gjennom hele LFS byggeprosessen. Den bør settes til navnet på mappen hvor du skal bygge LFS systemet ditt - vi vil bruke `/mnt/lfs` som et eksempel, men du kan velge et hvilket som helst mappenavn du ønsker. 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 denne variabelen satt er fordelaktig ved at kommandoer som f.eks `mkdir -v $LFS/tools` kan skrives bokstavelig. Skallet vil automatisk erstatte «`$LFS`» med «`/mnt/lfs`» (eller hvilken verdi variabelen ble satt til) når den behandler kommandolinjen.

Sett nå masken for opprettelse av filmodus (umask) til `022` i tilfelle vertsdistroen bruker en annen standard:

```
umask 022
```

Å sette `umask` til `022` sikrer at nyopprettede filer og mapper er skrivbare bare av eieren, men er lesbare og søkbar (bare for mapper) av alle (forutsatt at standardmoduser er brukt av `open(2)` systemkall, vil nye filer ende opp med tillatelsesmodus `644` og mapper med modus `755`). En overtillatt standard kan gi sikkerhetshull i LFS systemet, og en overbegrensende standard kan forårsake merkelige problemer under bygging eller bruk av LFS systemet.



Obs

Ikke glem å sjekke at `LFS` er satt og `umask` er satt til `022` 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 byggeplassering, som er `/mnt/lfs` hvis gitt eksempel ble fulgt.

Sjekk at umasken er satt opp riktig med:

```
umask
```

Utdataen kan være `0022` eller `022` (antall innledende nuller avhenger av vertsdistroen).

Hvis noen utdata fra disse to kommandoene er feil, bruk kommandoen gitt tidligere på denne siden for å sette `$LFS` til det riktige mappenavn og sett `umask` til `022`.



Notat

En måte å sikre at `LFS` variabelen og `umask` alltid er satt riktig er å redigere `.bash_profile` filen i både din personlige hjemmemappe og i `/root/.bash_profile` og skrive **export** og **umask** kommandoene over. I tillegg må skallet spesifisert i `/etc/passwd` filen for alle brukere som trenger `LFS` variabelen være `bash` for å sikre at `.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 kommandoen til filen `.bashrc` for brukeren og `root`. I tillegg, noen distribusjoner bruk en "if" test, og kjører ikke de resterende `.bashrc` instruksjoner for en ikke-interaktiv `bash` påkallelse. Pass på å plassere kommandoen foran testen for ikke interaktiv bruk.

2.7. Montering av den nye partisjonen

Nå som et filsystem er opprettet, må partisjonen monteres slik at vertssystemet kan få tilgang til det. Denne boken forutsetter at filsystemet er montert i katalogen spesifisert av `LFS` miljøvariabel beskrevet i forrige avsnitt.

Strengt tatt kan man ikke «montere en partisjon». Man monterer *filsystemet* innebygd i den partisjonen. Men siden en enkelt partisjon ikke kan inneholde mer enn ett filsystem, folk snakker ofte om partisjonen og tilhørende filsystem som om de var ett og samme.

Opprett monteringspunktet og monter `LFS` filsystemet med disse kommandoene:

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

Erstatt `<xxx>` med navnet for `LFS` partisjon.

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

```
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.

Angi eier- og tillatelsesmodus for `$LFS` mappen (dvs. rotmappen i det nyopprettede filsystemet for `LFS`-systemet) til `root` og `755` i tilfelle vertsdistroen er konfigurert til å bruke en annen standard for **mkfs**:

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

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, må partisjonene monteres på nytt.



Advarsel

Instruksjonene ovenfor forutsetter at du ikke starter datamaskinen på nytt din gjennom hele `LFS` prosessen. Hvis du slår av systemet, må du enten montere `LFS` partisjonen på nytt hver gang du starter byggeprosessen på nytt eller modifisere vertssystemets `/etc/fstab` filen til å automatisk monter den på nytt når du starter på nytt. Du kan for eksempel legge til denne linjen i `/etc/fstab` filen:

```
/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 den nye LFS partisjonen er klar til bruk, er det på tide å laste ned pakkene.

Kapittel 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 en 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 læsninger. Disse læsningene 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 eller til og med identiske filnavn. Men utgivelsens tarball kan inneholde noen filer som er viktige selv om de ikke er lagret i depotet (for eksempel a **configure** skript generert av **autoconf**), i tillegg til innholdet i tilsvarende øyeblikksbilde av depot. Boken bruker utgivelsens tarballer når det er mulig. Bruke et øyeblikksbilde av depot i stedet for en utgivelsens tarball spesifisert av boken vil forårsake problemer.

Nedlastingsplasseringer er kanskje ikke alltid tilgjengelige. Hvis en nedlastingsplasseringen har endret seg siden denne boken ble publisert, Google (<https://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 arbeidsmappe. 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 nedlastingsøkten :

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

Gjør denne mappen skrivbar og låst (sticky). «Sticky» betyr at selv om flere brukere har skrive tillatelse 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 for 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 wgetliste som beskrevet nedenfor.

For å laste ned alle pakkene og oppdateringene ved å bruke `wget-list-sysv` som en inngang til kommandoen **wget**, bruk:

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

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.

Hvis pakkene og oppdateringene er lastet ned som ikke-`root` bruker, vil disse filene eies av brukeren. Filsystemet registrerer eier ved hjelp av UID, og UID til en vanlig bruker i vertsdistroen er ikke tildelt i LFS. Så filene vil bli eid av en ikke navngitt UID i det endelige LFS systemet. Hvis du ikke vil tilordne samme UID for brukeren din i LFS systemet, endre eierne av disse filene til `root` nå for å unngå dette problemet:

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

3.2. Alle pakker



Notat

Les *sikkerhetsrådene* før du laster ned pakker for å finne ut om en nyere versjon av noen pakken bør brukes for å unngå sikkerhetssårbarheter.

Oppstrøms kilder kan fjerne gamle utgivelser, spesielt når de utgivelser inneholder en sikkerhetssårbarhet. Hvis en URL nedenfor ikke er tilgjengelig, bør du lese sikkerhetsrådene først for å finne ut om en nyere versjon (med sårbarheten fikset) skal brukes. Hvis ikke, prøv å laste ned den fjernede pakken fra et speil. Selv om det er mulig å laste ned en gammel utgivelse fra et speil selv om denne utgivelsen har blitt fjernet på grunn av en sårbarhet, er det ikke en god ide å bruk en utgivelse som er kjent for å være sårbar for å bygge systemet ditt.

Last ned eller på annen måte skaff deg følgende pakker:

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

Hjemmeside: <https://savannah.nongnu.org/projects/acl>

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

MD5 sum: 590765dee95907dbc3c856f7255bd669

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

Hjemmeside: <https://savannah.nongnu.org/projects/attr>

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

MD5 sum: 227043ec2f6ca03c0948df5517f9c927

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

Hjemmeside: <https://www.gnu.org/software/autoconf/>

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

MD5 sum: 1be79f7106ab6767f18391c5e22be701

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

Hjemmeside: <https://www.gnu.org/software/automake/>

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

MD5 sum: 7ab3a02318fee6f5bd42adfc369abf10

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

Hjemmeside: <https://www.gnu.org/software/bash/>

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

MD5 sum: 9c28f21ff65de72ca329c1779684a972

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

Hjemmeside: <https://git.gavinhoward.com/gavin/bc>

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

MD5 sum: ad4db5a0eb4fdbb3f6813be4b6b3da74

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

Hjemmeside: <https://www.gnu.org/software/binutils/>

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

MD5 sum: 49912ce774666a30806141f106124294

• **Bison (3.8.2) - 2,752 KB:**

Hjemmeside: <https://www.gnu.org/software/bison/>

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

MD5 sum: c28f119f405a2304ff0a7ccdccc629713

• **Bzip2 (1.0.8) - 792 KB:**

Nedlasting: <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>

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

MD5 sum: 50fcafccecd5a380415b12e9c574e0b2

• **Coreutils (9.6) - 5,991 KB:**

Hjemmeside: <https://www.gnu.org/software/coreutils/>

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

MD5 sum: 0ed6cc983fe02973bc98803155cc1733

• **DejaGNU (1.6.3) - 608 KB:**

Hjemmeside: <https://www.gnu.org/software/dejagnu/>

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

MD5 sum: 68c5208c58236eba447d7d6d1326b821

• **Diffutils (3.11) - 1,881 KB:**

Hjemmeside: <https://www.gnu.org/software/diffutils/>

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

MD5 sum: 75ab2bb7b5ac0e3e10cece85bd1780c2

• **E2fsprogs (1.47.2) - 9,763 KB:**

Hjemmeside: <https://e2fsprogs.sourceforge.net/>

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

MD5 sum: 752e5a3ce19aea060d8a203f2fae9baa

• **Elfutils (0.192) - 11,635 KB:**

Hjemmeside: <https://sourceware.org/elfutils/>

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

MD5 sum: a6bb1efc147302cfc15b5c2b827f186a

• **Expat (2.6.4) - 476 KB:**

Hjemmeside: <https://libexpat.github.io/>

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

MD5 sum: 101fe3e320a2800f36af8cf4045b45c7

• **Expect (5.45.4) - 618 KB:**

Hjemmeside: <https://core.tcl.tk/expect/>

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

MD5 sum: 00fce8de158422f5ccd2666512329bd2

• **File (5.46) - 1,283 KB:**

Hjemmeside: <https://www.darwinsys.com/file/>

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

MD5 sum: 459da2d4b534801e2e2861611d823864

• Findutils (4.10.0) - 2,189 KB:Hjemmeside: <https://www.gnu.org/software/findutils/>Nedlasting: <https://ftp.gnu.org/gnu/findutils/findutils-4.10.0.tar.xz>

MD5 sum: 870cfd71c07d37ebe56f9f4aaf4ad872

• Flex (2.6.4) - 1,386 KB:Hjemmeside: <https://github.com/westes/flex>Nedlasting: <https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz>

MD5 sum: 2882e3179748cc9f9c23ec593d6adc8d

• Flit-core (3.11.0) - 51 KB:Hjemmeside: <https://pypi.org/project/flit-core/>Nedlasting: https://pypi.org/packages/source/f/flit-core/flit_core-3.11.0.tar.gz

MD5 sum: 6d677b1acef1769c4c7156c7508e0dbd

• Gawk (5.3.1) - 3,428 KB:Hjemmeside: <https://www.gnu.org/software/gawk/>Nedlasting: <https://ftp.gnu.org/gnu/gawk/gawk-5.3.1.tar.xz>

MD5 sum: 4e9292a06b43694500e0620851762eec

• GCC (14.2.0) - 90,144 KB:Hjemmeside: <https://gcc.gnu.org/>Nedlasting: <https://ftp.gnu.org/gnu/gcc/gcc-14.2.0/gcc-14.2.0.tar.xz>

MD5 sum: 2268420ba02dc01821960e274711bde0

• GDBM (1.24) - 1,168 KB:Hjemmeside: <https://www.gnu.org/software/gdbm/>Nedlasting: <https://ftp.gnu.org/gnu/gdbm/gdbm-1.24.tar.gz>

MD5 sum: c780815649e52317be48331c1773e987

• Gettext (0.24) - 8,120 KB:Hjemmeside: <https://www.gnu.org/software/gettext/>Nedlasting: <https://ftp.gnu.org/gnu/gettext/gettext-0.24.tar.xz>

MD5 sum: 87aea3013802a3c60fa3feb5c7164069

• Glibc (2.41) - 18,892 KB:Hjemmeside: <https://www.gnu.org/software/libc/>Nedlasting: <https://ftp.gnu.org/gnu/glibc/glibc-2.41.tar.xz>

MD5 sum: 19862601af60f73ac69e067d3e9267d4

**Notat**

Glibc utviklerne opprettholder en *Git branch* som inneholder oppdateringer som anses verdig Glibc-2.41 men utviklet seg dessverre etter Glibc-2.41 utgivelsen. LFS redaksjonen vil utstede en sikkerhetsrådgivning hvis noen sikkerhetsfixes legges til i grenen, men ingen handlinger vil bli utført for andre oppdateringer som legges til. Du kan gjennomgå oppdateringene selv og inkludere noen oppdateringer hvis du anser dem som viktige.

• GMP (6.3.0) - 2,046 KB:Hjemmeside: <https://www.gnu.org/software/gmp/>Nedlasting: <https://ftp.gnu.org/gnu/gmp/gmp-6.3.0.tar.xz>

MD5 sum: 956dc04e864001a9c22429f761f2c283

• Gperf (3.1) - 1,188 KB:Hjemmeside: <https://www.gnu.org/software/gperf/>Nedlasting: <https://ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz>

MD5 sum: 9e251c0a618ad0824b51117d5d9db87e

• **Grep (3.11) - 1,664 KB:**

Hjemmeside: <https://www.gnu.org/software/grep/>

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

MD5 sum: 7c9bbd74492131245f7cdb291fa142c0

• **Groff (1.23.0) - 7,259 KB:**

Hjemmeside: <https://www.gnu.org/software/groff/>

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

MD5 sum: 5e4f40315a22bb8a158748e7d5094c7d

• **GRUB (2.12) - 6,524 KB:**

Hjemmeside: <https://www.gnu.org/software/grub/>

Nedlasting: <https://ftp.gnu.org/gnu/grub/grub-2.12.tar.xz>

MD5 sum: 60c564b1bdc39d8e43b3aab4bc0fb140

• **Gzip (1.13) - 819 KB:**

Hjemmeside: <https://www.gnu.org/software/gzip/>

Nedlasting: <https://ftp.gnu.org/gnu/gzip/gzip-1.13.tar.xz>

MD5 sum: d5c9fc9441288817a4a0be2da0249e29

• **Iana-Etc (20250123) - 591 KB:**

Hjemmeside: <https://www.iana.org/protocols>

Nedlasting: <https://github.com/Mic92/iana-etc/releases/download/20250123/iana-etc-20250123.tar.gz>

MD5 sum: f8a0ebdc19a5004cf42d8bdcf614fa5d

• **Inetutils (2.6) - 1,724 KB:**

Hjemmeside: <https://www.gnu.org/software/inetutils/>

Nedlasting: <https://ftp.gnu.org/gnu/inetutils/inetutils-2.6.tar.xz>

MD5 sum: 401d7d07682a193960bcdecafd03de94

• **Intltool (0.51.0) - 159 KB:**

Hjemmeside: <https://freedesktop.org/wiki/Software/intltool>

Nedlasting: <https://launchpad.net/intltool/trunk/0.51.0/+download/intltool-0.51.0.tar.gz>

MD5 sum: 12e517cac2b57a0121cda351570f1e63

• **IPRoute2 (6.13.0) - 906 KB:**

Hjemmeside: <https://www.kernel.org/pub/linux/utils/net/iproute2/>

Nedlasting: <https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-6.13.0.tar.xz>

MD5 sum: 1603d25120d03feeaba9b360d03ffaec

• **Jinja2 (3.1.5) - 239 KB:**

Hjemmeside: <https://jinja.palletsprojects.com/en/3.1.x/>

Nedlasting: <https://pypi.org/packages/source/J/Jinja2/jinja2-3.1.5.tar.gz>

MD5 sum: 083d64f070f6f1b5f75971ae60240785

• **Kbd (2.7.1) - 1,438 KB:**

Hjemmeside: <https://kbd-project.org/>

Nedlasting: <https://www.kernel.org/pub/linux/utils/kbd/kbd-2.7.1.tar.xz>

MD5 sum: f15673d9f748e58f82fa50cff0d0fd20

• **Kmod (34) - 331 KB:**

Hjemmeside: <https://github.com/kmod-project/kmod>

Nedlasting: <https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-34.tar.xz>

MD5 sum: 3e6c5c9ad9c7367ab9c3cc4f08dfde62

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

Hjemmeside: <https://www.greenwoodsoftware.com/less/>

Nedlasting: <https://www.greenwoodsoftware.com/less/less-668.tar.gz>

MD5 sum: d72760386c5f80702890340d2f66c302

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

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

MD5 sum: 7b078c594a77e0f9cd53a0027471c3bc

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

Hjemmeside: <https://sites.google.com/site/fullycapable/>

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

MD5 sum: 0e186df9de9b1e925593a96684fe2e32

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

Hjemmeside: <https://sourceware.org/libffi/>

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

MD5 sum: 696a1d483a1174ce8a477575546a5284

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

Hjemmeside: <https://libpipeline.nongnu.org/>

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

MD5 sum: 17ac6969b2015386bcb5d278a08a40b5

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

Hjemmeside: <https://www.gnu.org/software/libtool/>

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

MD5 sum: 22e0a29df8af5fdde276ea3a7d351d30

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

Hjemmeside: <https://github.com/besser82/libxcrypt/>

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

MD5 sum: 1796a5d20098e9dd9e3f576803c83000

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

Hjemmeside: <https://www.kernel.org/>

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

MD5 sum: 13b9e6c29105a34db4647190a43d1810



Notat

Linuxkjernen oppdateres ganske ofte, mange ganger pga oppdagelser av sikkerhetssårbarheter. Den siste tilgjengelige stabile kjerneversjonen kan bli 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.

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

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

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

MD5 sum: dead9f5f1966d9ae56e1e32761e4e675

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

Hjemmeside: <https://www.gnu.org/software/m4/>

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

MD5 sum: 0d90823e1426f1da2fd872df0311298d

• **Make (4.4.1) - 2,300 KB:**

Hjemmeside: <https://www.gnu.org/software/make/>

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

MD5 sum: c8469a3713cbb04d955d4ae4be23eeb

• **Man-DB (2.13.0) - 2,023 KB:**

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

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

MD5 sum: 97ab5f9f32914eef2062d867381d8cee

• **Man-pages (6.12) - 1,838 KB:**

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

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

MD5 sum: 44de430a598605eaba3e36dd43f24298

• **MarkupSafe (3.0.2) - 21 KB:**

Hjemmeside: <https://palletsprojects.com/p/markupsafe/>

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

MD5 sum: cb0071711b573b155cc8f86e1de72167

• **Meson (1.7.0) - 2,241 KB:**

Hjemmeside: <https://mesonbuild.com>

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

MD5 sum: c20f3e5ebbb007352d22f4fd6ceb925c

• **MPC (1.3.1) - 756 KB:**

Hjemmeside: <https://www.multiprecision.org/>

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

MD5 sum: 5c9bc658c9fd0f940e8e3e0f09530c62

• **MPFR (4.2.1) - 1,459 KB:**

Hjemmeside: <https://www.mpfr.org/>

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

MD5 sum: 523c50c6318dde6f9dc523bc0244690a

• **Ncurses (6.5) - 2,156 KB:**

Hjemmeside: <https://www.gnu.org/software/ncurses/>

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

MD5 sum: ac2d2629296f04c8537ca706b6977687

• **Ninja (1.12.1) - 235 KB:**

Hjemmeside: <https://ninja-build.org/>

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

MD5 sum: 6288992b05e593a391599692e2f7e490

• **OpenSSL (3.4.1) - 17,917 KB:**

Hjemmeside: <https://www.openssl-library.org/>

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

MD5 sum: fb7a747ac6793a7ad7118eaba45db379

• **Patch (2.7.6) - 766 KB:**

Hjemmeside: <https://savannah.gnu.org/projects/patch/>

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

MD5 sum: 78ad9937e4caadcba1526ef1853730d5

• **Perl (5.40.1) - 13,605 KB:**

Hjemmeside: <https://www.perl.org/>

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

MD5 sum: bab3547a5cdf2302ee0396419d74a42e

• **Pkgconf (2.3.0) - 309 KB:**

Hjemmeside: <https://github.com/pkgconf/pkgconf>

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

MD5 sum: 833363e77b5bed0131c7bc4cc6f7747b

• **Procps (4.0.5) - 1,483 KB:**

Hjemmeside: <https://gitlab.com/procps-ng/procps/>

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

MD5 sum: 90803e64f51f192f3325d25c3335d057

• **Psmisc (23.7) - 423 KB:**

Hjemmeside: <https://gitlab.com/psmisc/psmisc>

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

MD5 sum: 53eae841735189a896d614cba440eb10

• **Python (3.13.2) - 22,091 KB:**

Hjemmeside: <https://www.python.org/>

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

MD5 sum: 4c2d9202ab4db02c9d0999b14655dfe5

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

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

MD5 sum: d6aede88f480a018d26b3206f21654ae

• **Readline (8.2.13) - 2,974 KB:**

Hjemmeside: <https://tiswww.case.edu/php/chet/readline/rltop.html>

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

MD5 sum: 05080bf3801e6874bb115cd6700b708f

• **Sed (4.9) - 1,365 KB:**

Hjemmeside: <https://www.gnu.org/software/sed/>

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

MD5 sum: 6aac9b2dbafcd5b7a67a8a9bcb8036c3

• **Setuptools (75.8.1) - 1,313 KB:**

Hjemmeside: <https://pypi.org/project/setuptools/>

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

MD5 sum: 7dc3d3f529b76b10e35326e25c676b30

• **Shadow (4.17.3) - 2,274 KB:**

Hjemmeside: <https://github.com/shadow-maint/shadow/>

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

MD5 sum: 0da190e53ecee76237e4c8f3f39531ed

• **Sysklogd (2.7.0) - 465 KB:**

Hjemmeside: <https://www.infodrom.org/projects/sysklogd/>

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

MD5 sum: 611c0fa5c138eb7a532f3c13bdf11ebc

• **Systemd (257.3) - 15,847 KB:**

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

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

MD5 sum: 8e4fc90c7aead651fa5c50bd1b34abc2

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

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

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

MD5 sum: 9b77c3b066723d490cb10aed4fb05696



Notat

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

• **SysVinit (3.14) - 236 KB:**

Hjemmeside: <https://savannah.nongnu.org/projects/sysvinit>

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

MD5 sum: bc6890b975d19dc9db42d0c7364dd092

• **Tar (1.35) - 2,263 KB:**

Hjemmeside: <https://www.gnu.org/software/tar/>

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

MD5 sum: a2d8042658cfd8ea939e6d911eaf4152

• **Tcl (8.6.16) - 11,406 KB:**

Hjemmeside: <https://tcl.sourceforge.net/>

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

MD5 sum: eaef5d0a27239fb840f04af8ec608242

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

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

MD5 sum: 750c221bcb6f8737a6791c1fbe98b684

• **Texinfo (7.2) - 6,259 KB:**

Hjemmeside: <https://www.gnu.org/software/texinfo/>

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

MD5 sum: 11939a7624572814912a18e76c8d8972

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

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

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

MD5 sum: 404229390c06b7440f5e48d12c1a3251

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

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

MD5 sum: acd4360d8a5c3ef320b9db88d275dae6

• **Util-linux (2.40.4) - 8,641 KB:**

Hjemmeside: <https://git.kernel.org/pub/scm/utils/util-linux/util-linux.git/>

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

MD5 sum: f9cbb1c8315d8ccbeb0ec36d10350304

• **Vim (9.1.1166) - 18,077 KB:**

Hjemmeside: <https://www.vim.org>

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

MD5 sum: 718d43ce957ab7c81071793de176c2eb



Notat

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

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

Hjemmeside: <https://pypi.org/project/wheel/>

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

MD5 sum: dddc505d0573d03576c7c6c5a4fe0641

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

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

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

MD5 sum: 89a8e82cfd2ad948b349c0a69c494463

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

Hjemmeside: <https://tukaani.org/xz>

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

MD5 sum: 4b1cf07d45ec7eb90a01dd3c00311a3e

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

Hjemmeside: <https://zlib.net/>

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

MD5 sum: 9855b6d802d7fe5b7bd5b196a2271655

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

Hjemmeside: <https://facebook.github.io/zstd/>

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

MD5 sum: 780fc1896922b1bc52a4e90980cdda48

Total størrelse på disse pakkene: ca 527 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:

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

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

MD5 sum: 6a5ac7e89b791aae556de0f745916f7f

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

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

MD5 sum: 6aee45dd3e05b7658971c321d92f44b7

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

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

MD5 sum: 0b8b5ac411d011263ad40b0664c669f0

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

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

MD5 sum: 9a5997c3452909b1769918c759eff8a2

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

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

MD5 sum: f75cca16a38da6caa7d52151f7136895

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

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

MD5 sum: 3af8fd8e13cad481eeaa48be4247445

Total størrelse på disse oppdateringene: ca 190.7 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 anskaffe eventuelle tilleggoppdateringer som passer dine systembehov.

Kapittel 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` (hvor vi vil installere midlertidige verktøy, legg til en uprivilegert brukerå bli forklare tidsenhetene («SBU») som vi bruker til å måle hvor lang tid det tar å bygge LFS pakker, og gi litt informasjon om testpakkene til pakkene.

4.2. Opprette et begrenset mappeoppsett i LFS filsystemet

I denne delen begynner vi å fylle LFS filsystemet med deler som vil utgjøre det endelige Linuxsystemet. Det første trinnet er å opprette et begrenset kataloghierarki, slik at programmene som kompiles i Kapittel 6 (i tillegg til `glibc` og `libstdc++` i Kapittel 5) kan installeres i deres endelige plassering. Vi gjør dette slik at de midlertidige programmene vil bli overskrevet når de endelige versjonene bygges i Kapittel 8.

Create the required directory layout by issuing the following commands as `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 kan bli funnet i avsnitt Verktøykjedens tekniske merknader). Denne krysskompilatoren vil bli installert i en spesiell katalog for å skille den fra de andre programmene. Fortsatt som `root`, lag den mappen med denne kommandoen:

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



Notat

LFS redaksjonen har bevisst besluttet å ikke bruke en `/usr/lib64` mappe. Flere skritt tas for å være sikker på at verktøykjeden ikke vil bruke den. Hvis for noen grunn denne katalogen vises (enten fordi du gjorde en feil når du fulgte instruksjonene, eller fordi du installerte en binær pakke som opprettet det etter å ha fullført LFS), kan det ødelegge systemet ditt. Du bør alltid være sikker på at denne katalogen ikke eksisterer.

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 et rent arbeidsmiljø, opprett en ny bruker kalt `lfs` som medlem av en ny gruppe (også kalt `lfs`) og kjøre kommandoer som `lfs` 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
```

Dette er hva kommandolinjealternativene betyr:

```
-s /bin/bash
```

Dette gjør **bash** til 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 en skjelettmappe (standard er `/etc/skel`) ved å endre inndataplasingen til den spesielle nullenheten.

`lfs`

Dette er navnet til den nye brukeren.

Hvis du vil logge inn som `lfs` eller bytte til `lfs` fra en ikke-`root` bruker (i motsetning til å bytte til bruker `lfs` når du er logget inn som `root`, som ikke krever at `lfs` brukeren har et passord), må du angi et passord for `lfs`. Utsted følgende kommando som `root` bruker for å angi passordet:

```
passwd lfs
```

Bevilg `lfs` full tilgang til alle mapper under `$LFS` ved å gjøre `lfs` eieren:

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



Notat

I noen vertssystemer, følgende `su` kommando fullføres ikke riktig og suspenderer påloggingen for `lfs` bruker i bakgrunnen. Hvis ledeteksten "`lfs:~$`" ikke vises umiddelbart, å skrive inn `fg` kommandoen vil løse problemet.

Deretter starter du et skall som kjører som bruker `lfs`. Dette kan gjøres ved å logge inn som `lfs` på en virtuell konsoll, eller med følgende bytt ut/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 for 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 pålogget som bruker `lfs`, eller når du bytter til `lfs` bruker med en `su` kommando med «-» alternativet, det første skallet er et *login* skall som leser `/etc/profile` til 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 et 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.

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** sin 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 tilgjengelig uten å huske en tidligere versjon av det samme programmet levert av vertsdistroen, i `/usr/bin` eller `/bin`.

```
umask 022
```

Innstilling av `umask` som vi allerede har forklart i Seksjon 2.6, «Stille inn `$LFS` variabelen og `Umask`»

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

`LFS` variabelen skal settes til det valgte monteringspunktet.

```
LC_ALL=POSIX
```

`LC_ALL` variabelen 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
```

The `LFS_TGT` variabel setter en ikkestandard, men kompatibel maskinbeskrivelse for bruk når du bygger vår krysskompiler og linker og når du krysskompiler vår midlertidige verktøykjede. Mer informasjon finnes i Verktøykjedens tekniske merknader.

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

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

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

Hvis `/bin` ikke er en 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, for å gjøre dem synlige innenfor eventuelle underskall, eksporterer vi dem.



Viktig

Flere kommersielle distribusjoner legger til en ikke dokumentert instansiering av `/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 tilstedeværelse av `/etc/bash.bashrc` og flytt den hvis den er tilstede Som `root` bruker, gjør:

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

Når `lfs` brukeren ikke lenger er nødvendig (i begynnelsen av Kapittel 7), kan du trygt gjenopprette `/etc/bash.bashrc` (hvis ønsket).

Legg merke til at LFS Bash pakken vi bygger i Seksjon 8.36, «Bash-5.2.37» ikke er konfigurert til å laste eller kjøre `/etc/bash.bashrc`, så denne filen er ubrukelig på et fullført LFS system.

For mange moderne systemer med flere prosessorer (eller kjerner) kan kompileringstiden for en pakke reduseres ved å utføre en "parallell make" ved å fortelle make programmet hvor mange prosessorer som er tilgjengelige via et kommandolinjealternativ eller en miljøvariabel. For eksempel en Intel Core i9-13900K-prosessor har 8 P (ytelse) kjerner og 16 E (effektivitet) kjerner, og en P-kjerne kan kjøre to tråder samtidig så hver P-kjerne er modellert som to logiske kjerner av Linuxkjernen. Som et resultat er det totalt 32 logiske kjerner. En åpenbar måte å bruke alle disse logiske kjernene er å tillate **make** å gjøre opptil 32 byggejobber. Dette kan gjøres ved å sende `-j32` alternativet til **make**:

```
make -j32
```

Eller angi `MAKEFLAGS` miljøvariabel og dens innhold vil automatisk bli brukt av **make** som kommandolinjealternativer:

```
export MAKEFLAGS=-j32
```



Viktig

Aldri send en `-j` alternativ uten nummer til **make** eller angi et slikt alternativ i `MAKEFLAGS`. Å gjøre det vil tillate **make** å skape uendelige byggejobber og forårsake problemer med systemstabilitet.

For å bruke alle logiske kjerner som er tilgjengelige for å bygge pakker i Kapittel 5 og Kapittel 6, sett `MAKEFLAGS` nå i `.bashrc`:

```
cat >> ~/.bashrc << "EOF"
export MAKEFLAGS=-j$(nproc)
EOF
```

Erstatt `$(nproc)` med antallet logiske kjerner du vil bruke hvis du ikke vil bruke alle de logiske kjernene.

Til slutt, å ha miljøet fullt forberedt for å bygge midlertidige verktøy, tving **bash** skallet å lese den nye brukerprofilen:

```
source ~/.bash_profile
```

4.5. Om SBU

Mange vil gjerne vite på forhånd hvor lenge det tar å compilere 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 5 minutter på de raskeste systemene, men kan ta flere 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 kompiles fra denne boken er `binutils` i Kapittel 5. Tiden det tar å kompilere med en kjerne er det vi vil referere til som standard byggenhet eller SBU. Alle andre kompileringstider vil bli uttrykt i forhold til denne tidsenheten.

Tenk for eksempel på en pakke hvis kompileringstid er 4,5 SBU. Dette betyr at hvis et system tok 4 minutter å kompilere og installer det første passet med `binutils`, vil det ta *omtrent* 18 minutter å bygge denne eksempelpakken. Heldigvis er de fleste byggetidene kortere enn en SBU.

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.

På noen nyere systemer er hovedkortet i stand til å kontrollere systemets klokkehastighet. Dette kan styres med en kommando som f.eks `powerprofilesctl`. Dette er ikke tilgjengelig i LFS, men kan være tilgjengelig på vertsdistroen. Etter at LFS er fullført, kan det bli lagt til systemet med prosedyrene på *BLFS power-profiles-daemon* siden. Før du måler byggetiden til en pakke, anbefales det å bruke en systemeffektprofil satt for maksimal ytelse (og maksimal strømforbruk). Ellers kan den målte SBU verdien være unøyaktig fordi systemet kan reagere forskjellig når du bygger `binutils-pass1` eller andre pakker. Vær oppmerksom på at en betydelig unøyaktighet fortsatt kan dukke opp selv om den samme profilen brukes for begge pakkene fordi systemet kan reagere tregere hvis systemet er inaktivt når byggeprosedyren startes. Stille inn strømprofilen til «ytelse» vil minimere dette problemet. Og gjøre det vil selvsagt også gjøre at systemet bygger LFS raskere.

Hvis `powerprofilesctl` er tilgjengelig, utsted `powerprofilesctl set performance` kommandoen for å velge `performance` profilen. Noen distroer tilbyr `tuned-adm` kommandoen for å administrere profilene i stedet for `powerprofilesctl`, på disse distroene utsted `tuned-adm profile throughput-performance` kommandoen for å velge `throughput-performance` profilen.



Notat

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.

Tidene presentert her for alle pakker (unntatt `binutils-pass1` som er basert på én kjerne) er basert på bruk av fire kjerner (-j4). Tidene i kapittel 8 inkluderer også tiden det skal kjøres regresjonstestene for pakken med mindre annet er spesifisert.

4.6. Om testpakkene

De fleste pakkene gir en testpakke. å kjøre testpakken for en nybygd pakke er en god ide fordi den kan gi en «tilregnelighetssjekk» som indikerer at alt er compilert riktig. En testpakke som består kontrollene sine, 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.



Notat

Å kjøre testpakkene i Kapittel 5 og Kapittel 6 er meningsløst; siden testprogrammene er compilert med en krysskompilator, kan de sannsynligvis ikke 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 testpakker til en pakke 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/12.3/> for å bekrefte om disse feilene er forventet eller ikke. Denne siden er gyldig for alle tester i denne boken.

Del 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 til 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.



Viktig

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 bruke `tee` verktøyet for å sende terminalutdataen til en fil. Dette gjør feilsøkingen enklere hvis noe går galt.

Den neste delen er en teknisk introduksjon til byggeprosessen, mens den følgende presenterer **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, kompilasjonene i de resterende kapitlene vil være isolert 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 *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



Notat

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 er også referert til som «verten».

vert

er maskinen/systemet der de bygde programmene skal kjøres. Merk at denne bruken av «verten» 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 bygg og vertent.

Som et eksempel, la oss forestille oss følgende scenario (noen ganger referert til som «Canadian Cross»). vi har 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 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



Notat

Alle de krysskompilete pakkene i denne boken bruker en autoconf basert byggesystem. Det autoconf baserte byggesystemet godtar systemtyper i formen `cpu-vendor-kernel-os`, referert til som systemtripletten. Siden leverandørfeltet ofte er irrelevant, autoconf lar deg utelate det.

En klok leser kan lure på hvorfor en «triplett» refererer til et firekomponents navn. Kjernefeltet og os-feltet begynte som et enkelt «system» felt. Et slikt trefeltsskjema er fortsatt gyldig i dag for noen systemer, for eksempel, `x86_64-unknown-freebsd`. Men to systemer kan dele samme kjerne og fortsatt være for forskjellige for å bruke den samme tripletten for å beskrive dem. For eksempel Android kjørende på en mobiltelefon er helt forskjellig fra Ubuntu som kjører på en ARM64 server, selv om de begge kjører på samme type CPU (ARM64) og bruker samme kjerne (Linux).

Uten et emuleringslag kan du ikke kjøre en kjørbart fil for en server på en mobiltelefon eller omvendt. Så «system» feltet har blitt delt inn i kjerne- og os-felt, for å angi disse systemene entydig. I vårt eksempel, Android systemet er angitt `aarch64-unknown-linux-android`, og Ubuntu systemet er angitt `aarch64-unknown-linux-gnu`.

Ordet «triplett» forblir innebygd i leksikonet. En enkel måte å bestemme din systemtriplett er å kjøre **config.guess** skript som følger med kilden for mange pakker. Pakk ut `binutils` sine kilder, kjør skriptet `./config.guess`, og merk utdaten. 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`. På de fleste Linux systemer den enklere **gcc-dumpmachine** kommando vil gi deg lignende informasjon.

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 <navn på binær> | grep interpreter` og legg merke til utdataen. Den autoritative referansen som dekker alle plattformer er i *a Glibc wiki page*.

For å forfalske en krysskompilering i LFS, navnet på vertstripletten justeres litt ved å endre "vendor" feltet i `LFS_TGT` variabel så det står "lfs". 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 <code>cc1</code> ved å bruke <code>cc-pc</code> på <code>pc</code>
2	pc	lfs	lfs	Bygg kompilator <code>cc-lfs</code> ved å bruke <code>cc1</code> på <code>pc</code>

Stadie	Bygg	Vert	Mål	Handling
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.

Dette er ennå ikke slutten på historien. C-språket er ikke bare en kompilator; den definerer også et standardbibliotek. I denne boken er det GNU C-biblioteket, kalt glibc, som brukes (det finnes et alternativ, "musl"). Dette biblioteket må kompileres for LFS-maskinen; det vil si å bruke krysskompilatoren cc1. Men kompilatoren selv bruker et internt bibliotek som gir komplekse subrutiner for funksjoner som ikke er tilgjengelige i assembler-instruksjonssettet. Dette interne biblioteket heter libgcc, og det må være koblet til glibc for at biblioteket skal være fullt funksjonelt. Videre standardbiblioteket for C++ (libstdc++) må også være koblet til glibc. Løsningen på dette kylling og egg problemet er først å bygge en nedgradert cc1-basert libgcc, mangler noen funksjoner som tråder og unntakshåndtering, og da å bygge glibc ved å bruke denne nedgraderte kompilatoren (glibc selv er ikke nedgradert), og også for å bygge libstdc++. Dette siste biblioteket vil mangle noe av funksjonaliteten til libgcc.

Resultatet av det foregående avsnittet er at cc1 ikke er i stand til å bygge et fullt funksjonell libstdc++ med den nedgraderte libgcc, men cc1 er den eneste kompilatoren som er tilgjengelig for å bygge C/C++-bibliotekene under fase 2. Det er to grunner til at vi ikke umiddelbart bruker kompilator bygget i trinn 2, cc-lfs, for å bygge disse bibliotekene.

- Generelt sett kan ikke cc-lfs kjøre på pc (vertssystemet). Selv om tripllettene for pc og lfs er kompatible med hverandre, en kjørbart fil for lfs må avhenge av glibc-2.41; vertsdistroen kan bruke en annen implementering av libc (for eksempel musl), eller en tidligere utgivelse av glibc (for eksempel, glibc-2.13).
- Selv om cc-lfs kan kjøre på pc, vil det å bruke det på pc skape en risiko for å koble til pc-bibliotekene, siden cc-lfs er en lokal kompilator.

Så når vi bygger gcc trinn 2, instruerer vi byggesystemet til å gjenoppbygge libgcc og libstdc++ med cc1, men vi kobler libstdc++ til den nye gjenoppbygde libgcc i stedet for den gamle, nedgraderte konstruksjonen. Dette gjør den ombygde libstdc++ fullt funksjonell.

I Kapittel 8 (eller «stage 3»), alle pakkene som trengs for LFS systemet er bygget. Selv om en pakke allerede er installert i LFS systemet i et tidligere kapittel, bygger vi fortsatt pakken på nytt. Hovedårsaken til å gjenoppbygge disse pakkene er å gjøre dem stabile: hvis vi installerer en LFS pakke på nytt på et fullført LFS system, det reinstallerer innholdet i pakken skal være det samme som innholdet i den samme pakken når den først installeres i Kapittel 8. De midlertidige pakkene installert i Kapittel 6 eller Kapittel 7 kan ikke tilfredsstille dette kravet, fordi noen av dem er bygget uten valgfrie avhengigheter, og autoconf kan ikke utføre noen funksjonsinnsjekker Kapittel 6 på grunn av krysskompilering, forårsaker at de midlertidige pakkene mangler valgfrie funksjoner, eller bruker suboptimale koderutiner. I tillegg en mindre grunn til gjenoppbygging av pakkene er å kjøre testpakkene.

Andre prosedyredetaljer

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

Binutils installeres først fordi **configure** kjøring 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/$LFS_TGT/bin`. Verktøyene i en plassering er hardlenket til den andre. En viktig fasett av lenkeren er bibliotekets søkerekkfølge. Detaljert informasjon kan fås fra `ld` ved å gi den `--verbose` flagget. For eksempel, `$LFS_TGT-ld --verbose | grep SEARCH` vil illustrere gjeldende søkestier og rekkefølgen deres. (Merk at dette eksempelet kan kjøres som vist kun mens du er logget på som bruker `lfs`. Hvis du kommer tilbake til denne siden senere, bytt ut `$LFS_TGT-ld` med `ld`).

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 konfigureringskript ikke søker i STI (PATH) mapper for å finne hvilke verktøy det skal bruke. Imidlertid under selve kjøringen av `gcc` er ikke de samme søkestiene nødvendigvis brukt. For å finne ut hvilke standardlenker `gcc` vil bruke, kjør: `$LFS_TGT-gcc -print-prog-name=ld`. (En gang til, fjern `$LFS_TGT-` prefikset hvis du kommer tilbake til dette seinere.)

Detaljert informasjon kan fås fra `gcc` med å gi alternativet `-v` på kommandolinjen under kompilering av et program. For eksempel, `$LFS_TGT-gcc -v example.c` (eller uten `$LFS_TGT-` hvis du kommer tilbake senere) vises detaljert informasjon om forprosessoren, kompileringen og sammenstillings stadier, inkludert `gcc` sine søkestier for inkluderte deklarasjoner og deres rekkefølge.

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

Neste kommer glibc. Det viktigste hensynet for å bygge glibc er kompilatoren, binære verktøy og kjernedeklarasjoner. Kompilatoren og binære verktøy er generelt ikke et problem siden glibc alltid vil bruke kompilatoren som er relatert til `--host` parameteret sendt til konfigureringskriptet; f.eks. i vårt tilfelle vil kompilatoren være `$LFS_TGT-gcc` og `readelf` verktøyet vil være `$LFS_TGT-readelf`. Kjerne deklarasjonene kan være litt mer komplisert. Derfor tar vi ingen risiko og bruker den tilgjengelige konfigurasjonsbryteren for å fremtvinge riktig valg. Etter kjøring av `configure`, sjekk innholdet i `config.make` filen i `build` mappen for alle viktige detaljer. Disse elementene fremhever et viktig aspekt ved glibc pakken—en 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 andre programmer som må krysskompileres for å bryte sirkulære avhengigheter på byggetidspunktet. Installasjonstrinnet for alle disse pakkene bruker `DESTDIR` variabel for å tvinge installasjonen inn i LFS filsystemet.

Ved slutten av Kapittel 6 den lokale `lfs` kompilatoren er installert. Første `binutils-pass2` blir bygget, med den samme `DESTDIR` mappen 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 konfigureringskript, `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 erklært eksplisitt som et av konfigurasjonsalternativene.

Når du kommer inn i `chroot`-miljøet i Kapittel 7, de midlertidige installasjonene av programmer som trengs for riktig betjening av verktøykjeden utføres. Fra dette tidspunktet og fremover er kjerneverktøykjeden selvstendig og selvhostet. I Kapittel 8, endelige versjoner av alle pakker som trengs for et fullt funksjonelt system bygges, testes og installert.

Generelle kompileringsinstruksjoner



Obs

Under en utviklingsyklus av LFS er instruksjonene i boken ofte modifisert for å tilpasse seg en pakkeoppdatering eller dra nytte av nye funksjoner fra oppdaterte pakker. Å blande sammen instruksjonene til forskjellige versjoner av LFS boken kan forårsake subtile brudd. Denne typen problem er vanligvis et resultat av gjenbruk av et eller annet opprettet skript for en tidligere LFS utgivelse. Slik gjenbruk frarodes sterkt. Hvis du gjenbruker skript for en tidligere LFS utgivelse av en eller annen grunn, må du være veldig forsiktig med å oppdatere skriptene for å matche gjeldende versjon av LFS boken.

Her er noen ting du bør vite om å bygge hver pakke:

- Flere pakker oppdateres før kompilering, men bare den oppdateringen er nødvendig for å omgå et problem. En oppdatering er ofte nødvendig i både gjeldende og følgende kapitler, men noen ganger, når den samme pakken er bygget mer enn en gang, er ikke oppdateringen nødvendig med en gang. 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, noen advarsler vil rulle forbi på skjermen. Disse er normale og kan trygt bli ignorert. Disse advarslene handler vanligvis om utdatert, men ikke ugyldig, bruk av C- eller C++-syntaksen. C-standardene endres ganske ofte, og noen pakker er ennå ikke oppdatert. Dette er ikke et alvorlig problem, men det fører til at advarslene vises.
- 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:



Viktig

Byggeinstruksjonene forutsetter at Systemkrav for verten, 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.



Viktig

Her er en oversikt over byggeprosessen.

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 `/mnt/lfs/sources/` mappen.
3. For hver pakke:
 - a. Bruk **tar** programmet, pakke ut pakken som skal bygges. I Kapittel 5 og Kapittel 6, sikre at du er *lfs* brukeren når du pakker ut pakken.

Ikke bruk noen metode bortsett fra **tar** kommandoen for å trekke ut kildekoden. Spesielt ved å bruke **cp -R** kommandoen for å kopiere kildekode til et annet sted kan ødelegge tidsstempler i kildetreet, og føre til at byggingen mislykkes.

- 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 når byggingen er ferdig.
- e. Slett den utpakkede kildemappen med mindre du blir bedt om noe annet.

Kapittel 5. Kompilere en kryssverktøykjede

5.1. Introduksjon

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

Programmene som er kompilert i dette kapitlet 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-2.44 - Pass 1

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

Omtrentlig byggetid: 1 SBU
Nødvendig diskplass: 677 MB

5.2.1. Installasjon av Kryss Binutils



Notat

Gå tilbake og les notatene i avsnittet med tittelen Generelle kompileringsinstruksjoner. 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
```



Notat

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 \
             --enable-gprofng=no \
             --disable-werror \
             --enable-new-dtags \
             --enable-default-hash-style=gnu
```

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.

`--enable-gprofng=no`

Dette deaktiverer bygging av `gprofng` som ikke er nødvendig for midlertidige verktøy.

`--disable-werror`

Dette forhindrer byggingen i å stoppe i tilfelle det er advarsler fra vertens kompilator.

`--enable-new-dtags`

Dette gjør at linkerens bruker «runpath» taggen for å bygge inn biblioteksøkestier i kjørbare filer og delte biblioteker, i stedet for det tradisjonelle «rpath» taggen. Det gjør feilsøking av dynamisk koblede kjørbare filer enklere og fungerer rundt potensielle problemer i testpakken til enkelte pakker.

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

Som standard vil linkerens generere både GNU-stil hash tabell og den klassiske ELF-hash tabellen for delte biblioteker og dynamisk koblede kjørbare filer. Hash tabellene er kun ment for en dynamisk linker for å utføre symboloppslag. På LFS er dynamikken at linkerens (levert av Glibc-pakken) vil alltid bruke GNU-stil hashtabell som er raskere å spørre. Så klassikeren ELF hash-tabell er helt ubrukelig. Dette gjør at linkerens generer bare hashtabellen i GNU-stil som standard, slik at vi kan unngå å kaste bort tid på å generere den klassiske ELF-hash-tabellen når vi bygger pakkene, eller kaster bort diskplass for å lagre den.

Fortsett med å compilere pakken:

```
make
```

Installer pakken:

```
make install
```

Detaljer om denne pakken finner du i Seksjon 8.20.2, «Innhold i Binutils»

5.3. GCC-14.2.0 - Pass 1

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

Omtrentlig byggetid: 3.2 SBU

Nødvendig diskplass: 4.8 GB

5.3.1. Installasjon av Cross 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 byggeprosedyrene automatisk bruker dem:



Notat

Det er hyppige misforståelser om dette kapittelet. Prosedyrene er de samme som alle andre kapitler som forklart tidligere (Package build instructions). Først pakker du ut gcc-14.2.0 tarballen fra kildemappen, og endre deretter til den opprettede mappen. Først da bør du fortsett med instruksjonene nedenfor.

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

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=2.41 \
  --with-sysroot=$LFS \
  --with-newlib \
  --without-headers \
  --enable-default-pie \
  --enable-default-ssp \
  --disable-nls \
  --disable-shared \
  --disable-multilib \
  --disable-threads \
  --disable-libatomic \
  --disable-libgomp \
  --disable-libquadmath \
  --disable-libssp \
  --disable-libvtv \
  --disable-libstdcxx \
  --enable-languages=c,c++
```

Betydningen av konfigurasjonsalternativene:

```
--with-glibc-version=2.41
```

Dette alternativet spesifiserer versjonen av glibc som vil bli 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 tverrkompiletor, krever GCC standarddeklarasjoner som er kompatible med målsystemet. For vårt formål vil disse deklarasjonene ikke være nødvendige. Denne bryteren hindrer GCC i å lete etter dem.

```
--enable-default-pie and --enable-default-ssp
```

Disse bryterne lar GCC kompilere programmer med noen herdende sikkerhetsfunksjoner (mer informasjon om de i note on PIE and SSP kapittel 8) som standard. De er strengt tatt ikke nødvendig på dette stadiet, siden kompilatoren bare vil produsere midlertidige kjørbare filer. Men det er renere å ha de midlertidige pakkene så nær de endelige som mulig.

```
--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 ikke en multilib konfigurasjon. Denne bryteren er ufarlig for x86.

```
--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 krysskompiletor og er ikke nødvendig for oppgaven med å krysskompilere den midlertidige libc.

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

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

Kompiler GCC ved å kjøre:

```
make
```

Installer pakken:

```
make install
```

Dette bygget av GCC har installert et par interne systemdeklarasjoner. Normalt vil en av dem, `limits.h`, i sin tur inkludere den tilsvarende system `limits.h` systemdeklarasjonen, 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 deklarasjonen som nettopp har blitt installert er en delvis, selvstendig fil og inkluderer ikke de utvidede funksjonene til systemdeklarasjonen. Dette er tilstrekkelig for å bygge glibc, men den fullstendige interne deklarasjonen vil være nødvendig senere. Lag en fullversjon av den interne deklarasjonen ved å bruke en kommando som er identisk med det GCC byggesystemet gjør under normale omstendigheter:



Notat

Kommandoen nedenfor viser et eksempel på nestet kommandoerstatning ved å bruke to metoder: backquotes og a `$()` konstruksjon. Det kan skrives om ved å bruke samme metode for begge erstatningene, men vises på denne måten for å demonstrere hvordan de kan blandes. Som regel er `$()` metoden foretrukket.

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

Detaljer om denne pakken finner du i Seksjon 8.29.2, «Innhold i GCC»

5.4. Linux-6.13.4 API Deklarasjoner

Linux API deklarasjonene (i linux-6.13.4.tar.xz) eksponerer kjernens API for bruk av Glibc.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 1.6 GB

5.4.1. Installasjon av Linux API deklarasjoner

Linux-kjernen må eksponere et applikasjonsprogrammeringsgrensesnitt (Application Programming Interface(API)) som systemets C bibliotek (Glibc i LFS) kan bruke. Dette har blitt gjort ved å rense ulike C deklarasjonsfiler som er i Linux sin 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 make målet «headers_install» kan ikke brukes, fordi det krever rsync, som kanskje ikke er tilgjengelig. Deklarasjonene plasseres først i ./usr, deretter kopiert til nødvendig plassering.

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

5.4.2. Innhold i Linux API deklarasjoner

Installerte deklarasjoner: /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-2.41

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: 1.4 SBU
Nødvendig diskplass: 850 MB

5.5.1. Installasjon av Glibc

Først oppretter du en symbolsk lenke for LSB kompatibilitet. 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
```



Notat

Kommandoen ovenfor er riktig. **ln** kommandoen har flere syntaktiske versjoner, så sørg for å sjekke **info coreutils ln** og *ln(1)* før du rapporterer hva som kan se ut til å være en feil.

Noen Glibc programmer bruker den FHS inkompatible `/var/db` mappen for å lagre deres kjøretidsdata. Bruk følgende oppdatering for å få slike programmer til å lagre sine kjøretidsdata på FHS compatible steder:

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

Neste, forbered Glibc for kompilering:

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

Betydningen av konfigureringsalternativene:

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

Den kombinerte effekten av disse bryterne er at Glicbs byggesystem konfigurerer seg selv til å være krysskompilert, ved hjelp av krysskoblingen og krysskompilator i `$LFS/tools`.

```
--enable-kernel=5.4
```

Dette forteller Glibc å kompilere biblioteket med støtte for 5.4 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.

```
--disable-nscd
```

Ikke bygg navnetjenesten cache daemon som ikke er brukt lenger.

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.



Notat

Det har vært rapporter om at denne pakken kan mislykkes når den bygges som en «parallel make». Hvis dette skjer, kjør make kommandoen på nytt med et `-j1` alternativ.

Kompiler pakken:

```
make
```

Installer pakken:



Advarsel

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

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

Betydningen av make install alternativet:

```
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 `root (/)` mappen. Her spesifiserer vi at pakken installeres i `$LFS`, which will become the root directory in Seksjon 7.4, «Gå inn i Chroot miljøet».

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

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



Obs

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(){}' | $LFS_TGT-gcc -xc -
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 korrigere det. Dette problemet må løses før du fortsetter.

Når alt er bra, rydd opp i testfilene:

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



Notat

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

Detaljer om denne pakken finner du i Seksjon 8.5.3, «Innhold i Glibc»

5.6. Libstdc++ fra GCC-14.2.0

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 Libstdc++ avhenger av Glibc, som ennå ikke var tilgjengelig i målmappen.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 850 MB

5.6.1. Installasjon av Target Libstdc++



Notat

Libstdc++ er en del av GCC kildene. Du bør først pakke ut GCC tarball og bytte til `gcc-14.2.0` 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++/14.2.0
```

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 forhåndskompilerte include filer som ikke er nødvendige på dette stadiet.

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

Dette spesifiserer installasjonsmappen for include filer. Fordi libstdc++ er standard C++ biblioteket for LFS, skal denne mappen samsvare med plasseringen der C++ kompilatoren (`$LFS_TGT-g++`) vil søke etter standard C++ include filer. I en normal konstruksjon, sendes denne informasjonen automatisk til libstdc++ **configure** alternativer fra toppnivåmappen. I vårt tilfelle, denne informasjonen må gis eksplisitt. C++ kompilatoren vil legge til sysroot banen `$LFS` (spesifisert når GCC pass 1 ble bygget) til å inkludere filsøkebanen, så den vil faktisk søke i `$LFS/tools/$LFS_TGT/include/c++/14.2.0`. Kombinasjonen av `DESTDIR` variabelen (i **make install** kommandoen nedenfor) og denne bryteren sørger for å installere deklarasjonene der.

Kompiler libstdc++ ved å kjøre:

```
make
```

Installer biblioteket:

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

Fjern libtool arkivfilene fordi de er skadelige for krysskompilering:

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

Detaljer om denne pakken finner du i Seksjon 8.29.2, «Innhold i GCC»

Kapittel 6. Krysskompilering av midlertidige verktøy

6.1. Introduksjon

Dette kapitlet 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 kapitlet må gjøres som bruker `lfs`, med miljøet som beskrevet i Seksjon 4.4, «Sette opp miljøet»

6.2. M4-1.4.19

M4 pakken inneholder en makroprosessor.

Omtrentlig byggetid: 0.1 SBU

Nødvendig diskplass: 32 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 Seksjon 8.13.2, «Innhold i M4»

6.3. Ncurses-6.5

Ncurses pakken inneholder biblioteker for terminaluavhengig håndtering av karakterskjermer.

Omtrentlig byggetid: 0.4 SBU
Nødvendig diskplass: 53 MB

6.3.1. Installasjon av Ncurses

Først, kjør følgende kommandoer for å bygge «tic» programmet på byggeverten:

```
mkdir build
pushd build
  ./configure AWK=gawk
  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-normal \
  --with-cxx-shared \
  --without-debug \
  --without-ada \
  --disable-stripping \
  AWK=gawk
```

Betydningen av de nye konfigureringsalternativene:

--with-manpage-format=normal

Dette forhindrer at Ncurses installerer komprimerte manualsider, noe som kan skje hvis selve vertsdistribusjonen har komprimerte manualsider.

--with-shared

Dette får Ncurses til å bygge og installere delte C biblioteker.

--without-normal

Dette forhindrer at Ncurses bygger og installerer statiske C biblioteker.

--without-debug

Dette forhindrer at Ncurses bygger og installerer feilsøkingbiblioteker.

--with-cxx-shared

Dette får Ncurses til å bygge og installere delte C++ bindinger. Den forhindrer også at den bygger og installerer statiske C++ bindinger.

--without-ada

Dette sikrer at Ncurses ikke bygger støtte for Ada kompilatoren 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 fra å bruke **strip** programmet fra verten. Bruk av vertsverktøy på krysskompilerte programmer kan forårsake feil.

AWK=gawk

Denne bryteren hindrer byggesystemet å bruke **mawk** programmet fra verten. Noen versjoner av **mawk** kan gjøre at denne pakken ikke blir bygget.

Kompiler pakken:

```
make
```

Installer pakken:

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

Betydningen av installasjonsalternativene:

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

Vi må sende stien til den nettopp bygde **tic** programmet som kjører på byggemaskinen, slik at terminaldatabasen kan opprettes uten feil.

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

`libncurses.so` biblioteket trengs av noen få pakker vi skal bygge snart. Vi lager denne symbolkoblingen for å bruke `libncursesw.so` som en erstatning.

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

Deklarasjonsfilen `curses.h` inneholder definisjonen av ulike `Ncurses` datastrukturer. Med forskjellige preprocessor makro definisjoner to forskjellige sett med data strukturdefinisjon kan brukes: 8-biters definisjon er kompatibel med `libncurses.so` og definisjon av wide-character er kompatibel med `libncursesw.so`. Siden vi bruker `libncursesw.so` som erstatning for `libncurses.so`, rediger deklarasjonsfilen slik at den alltid vil bruke datastrukturdefinisjonen med wide-character kompatibel med `libncursesw.so`.

Detaljer om denne pakken finner du i Seksjon 8.30.2, «Innhold i `Ncurses`»

6.4. Bash-5.2.37

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

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 68 MB

6.4.1. Installasjon av Bash

Forbered Bash for kompilering:

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

Betydningen av konfigureringsalternativene:

--without-bash-malloc

Dette alternativet slår av bruken av Bash minnetildelingsfunksjon (`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 Seksjon 8.36.2, «Innholdet i Bash»

6.5. Coreutils-9.6

Coreutils pakken inneholder de grunnleggende hjelpeprogrammene som trengs av hvert operativsystem.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 181 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
```

Install the package:

```
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 Seksjon 8.58.2, «Innhold i Coreutils»

6.6. Diffutils-3.11

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

Omtrentlig byggetid: 0.1 SBU

Nødvendig diskplass: 35 MB

6.6.1. Installasjon av Diffutils

Forbered Diffutils 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 Seksjon 8.60.2, «Innhold i Diffutils»

6.7. File-5.46

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

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 42 MB

6.7.1. Installasjon av File

file kommandoen på byggevertens må være samme versjon som den vi bygger for å opprette signaturfilen. Kjør følgende kommandoer for å lage en midlertidig kopi av **file** kommandoen:

```
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
```

Fjern libtool arkivfilen fordi den er skadelig for krysskompilering:

```
rm -v $LFS/usr/lib/libmagic.la
```

Detaljer om denne pakken finner du i Seksjon 8.11.2, «Innholdet i File»

6.8. Findutils-4.10.0

Findutils pakken inneholder programmer for å finne filer. Programmer er tilgjengelig for å søke gjennom alle filene i et katalogtre og til opprette, vedlikeholde og søke i en database (ofte raskere enn den rekursive find, men upålitelig med mindre databasen nylig har blitt oppdatert). Findutils leverer også **xargs** programmet, som kan brukes til å kjøre en spesifisert kommando på hver fil valgt av et søk.

Omtrentlig byggetid: 0.2 SBU

Nødvendig diskplass: 48 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 Seksjon 8.62.2, «Innhold i Findutils»

6.9. Gawk-5.3.1

Gawk pakken inneholder programmer for å manipulere tekstfiler.

Omtrentlig byggetid: 0.1 SBU

Nødvendig diskplass: 47 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 Seksjon 8.61.2, «Innhold i Gawk»

6.10. Grep-3.11

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

Omtrentlig byggetid: 0.1 SBU

Nødvendig diskplass: 27 MB

6.10.1. Installasjon av Grep

Forbered Grep 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 Seksjon 8.35.2, «Innhold i Grep»

6.11. Gzip-1.13

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 Seksjon 8.65.2, «Contents of Gzip»

6.12. Make-4.4.1

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

Omtrentlig byggetid: less than 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 Seksjon 8.69.2, «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 Seksjon 8.70.2, «Innhold i oppdateringen»

6.14. Sed-4.9

Sed pakken inneholder en dataflyt (stream) redigerer.

Omtrentlig byggetid: 0.1 SBU

Nødvendig diskplass: 21 MB

6.14.1. Installasjon av Sed

Forbered Sed 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 Seksjon 8.31.2, «Innhold i Sed»

6.15. Tar-1.35

Tar pakken gir muligheten til å lage tar arkiver og 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 er lagret.

Omtrentlig byggetid: 0.1 SBU

Nødvendig diskplass: 42 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 Seksjon 8.71.2, «Innhold i Tar»

6.16. Xz-5.6.4

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

Omtrentlig byggetid: 0.1 SBU

Nødvendig diskplass: 21 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-5.6.4
```

Kompiler pakken:

```
make
```

Installer pakken:

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

Fjern libtool-arkivfilen fordi den er skadelig for krysskompilering:

```
rm -v $LFS/usr/lib/liblzma.la
```

Detaljer om denne pakken finner du i Seksjon 8.8.2, «Innhold i Xz»

6.17. Binutils-2.44 - Pass 2

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

Omtrentlig byggetid: 0.4 SBU
Nødvendig diskplass: 539 MB

6.17.1. Installation of Binutils

Binutils byggesystem er avhengig av en medsendt libtool kopi å lenke mot interne statiske biblioteker, men liberty- og zlib-kopiene sendt i pakken bruker ikke libtool. Denne inkonsekvensen kan forårsake at produserte binærfiler feilaktig kobler mot biblioteker fra vertens distro. Omgå dette problemet:

```
sed '6031s/$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 \
--enable-gprofng=no \
--disable-werror \
--enable-64-bit-bfd \
--enable-new-dtags \
--enable-default-hash-style=gnu
```

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
```

Fjern libtool arkivfilene fordi de er skadelige for krysskompilering, og fjern unødvendige statiske biblioteker:

```
rm -v $LFS/usr/lib/lib{bfd,ctf,ctf-nobfd,opcodes,sframe}.{a,la}
```

Detaljer om denne pakken finner du i Seksjon 8.20.2, «Innhold i Binutils»

6.18. GCC-14.2.0 - Pass 2

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

Omtrentlig byggetid: 4.1 SBU

Nødvendig diskplass: 5.5 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-4.2.1.tar.xz
mv -v mpfr-4.2.1 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.3.1.tar.gz
mv -v mpc-1.3.1 mpc
```

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
```

Overstyr byggerregelen for libgcc og libstdc++ deklarasjoner, til å tillate byggingen av disse bibliotekene med støtte for POSIX-tråder:

```
sed '/thread_header =/s/@.*@/gthr-posix.h/' \
-i libgcc/Makefile.in libstdc++-v3/include/Makefile.in
```

Opprett en egen byggemappe igjen:

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

Før du begynner å bygge GCC, husk å deaktivere alle miljøvariabler som overstyrrer standard optimaliseringsflagg.

Forbered nå GCC for kompilering:

```
../configure \
--build=$(./config.guess) \
--host=$LFS_TGT \
--target=$LFS_TGT \
LDFLAGS_FOR_TARGET=-L$PWD/$LFS_TGT/libgcc \
--prefix=/usr \
--with-build-sysroot=$LFS \
--enable-default-pie \
--enable-default-ssp \
--disable-nls \
--disable-multilib \
--disable-libatomic \
--disable-libatomic \
--disable-libatomic \
--disable-libatomic \
--disable-libatomic \
--disable-libatomic \
--disable-libatomic \
--disable-libatomic \
--disable-libatomic \
--disable-libatomic \
--disable-libatomic \
--enable-languages=c,c++
```

Betydningen av de nye konfigureringsalternativene:

`--with-build-sysroot=$LFS`

Normalt, å bruke `--host` sørger for at en krysskompilator brukes til å bygge GCC, og da vet denne 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.

```
--target=$LFS_TGT
```

Vi krysskompilerer GCC, så det er umulig å bygge målbibliotekene (`libgcc` og `libstdc++`) med GCC binærfiler kompilert i dette passet—disse binærfile vil ikke kjøre på verten. GCC byggesystemet vil forsøke å bruke verten's C og C++ kompilatorer som en standard løsning. Å bygge GCC målbibliotekene med en annen versjonen av GCC støttes ikke, så bruk av vertens kompilatorer kan føre til at byggingen mislykkes. Denne parameteren sikrer at bibliotekene bygges av GCC pass 1.

```
LDFLAGS_FOR_TARGET=...
```

Tillat `libstdc++` å bruke `libgcc` som ble bygget i dette passet, i stedet for den forrige versjonen innebygd i `gcc-pass1`. Den forrige versjonen kan ikke støtte C++ unntakshåndtering på riktig måte fordi den ble bygget uten `libc` støtte.

```
--disable-libsanitizer
```

Deaktiver GCC rensende kjøretidsbiblioteker. De er ikke nødvendig for den midlertidige installasjonen. I `gcc-pass1` det ble antydnet av `--disable-libstdcxx`, og nå kan vi gi den eksplisitt.

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. Å 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 Seksjon 8.29.2, «Innhold i GCC»

Kapittel 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 de såkalte *Virtuelle kjernefilssystemer (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** kommandoen.

Inntil Seksjon 7.4, «Gå inn i Chroot miljøet», må kommandoene kjøres som `root`, med `LFS` variabelen satt. 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 dårlige kommandoer.

7.2. Skifte eierskap



Notat

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

7.3. Forberede det virtuelle kjernefilsystemet

Applikasjoner som kjører i brukerområdet bruker forskjellige filsystemer opprettet av kjernen for å kommunisere med selve kjernen. Disse filsystemene er virtuelle: ingen diskplass brukes til dem. Innholdet i disse filsystemene ligger i minnet. Disse filsystemene må monteres i `$LFS` katalogtreet slik at applikasjonene kan finne dem i chroot miljøet.

Begynn med å lage mappene som disse virtuelle filsystemene vil være montert på:

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

7.3.1. Montering og fylling av /dev

Under en normal oppstart av et LFS systemet vil kjernen automatisk montere `devtmpfs` filsystemet på `/dev` mappen; kjernen oppretter enhetsnoder på det virtuelle filsystemet under oppstartsprosessen, eller når en enhet først oppdages eller åpnes. `udev` nissen kan endre eierskapet eller tillatelsene til enhetsnodene opprettet av kjernen, og lage

nye enhetsnoder eller symbolkoblinger for å lette arbeidet til distro vedlikeholdere og systemadministratorer. (Se Seksjon 9.3.2.2, «Oppretting av enhetsnode» for detaljer.) Hvis vertskjernen støtter `devtmpfs`, kan vi enkelt montere en `devtmpfs` på `$LFS/dev` og stole på at kjernen fyller den.

Men noen vertskjerner mangler `devtmpfs` støtte; disse vertsdistroene bruker forskjellige metoder for å lage innholdet i `/dev`. Så den eneste vert-agnostiske måten å fylle `$LFS/dev` mappen er ved å bind-montere vertssystemets `/dev` mappe. En bind-montering er en spesiell type montering som lager et mappeundertre eller en fil synlig på et annet sted. Bruk følgende kommando for å gjøre dette.

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

7.3.2. Montering av det virtuelle kjernefilssystemer

Monter nå de gjenværende virtuelle kjernefilssystemene:

```
mount -vt devpts devpts -o gid=5,mode=0620 $LFS/dev/pts
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
mount -vt tmpfs tmpfs $LFS/run
```

Betydningen av monteringsalternativene for `devpts`:

gid=5

Dette sikrer at alle `devpts` opprettede enhetsnoder eies av gruppe ID 5. Dette er IDen vi skal bruke senere for `tty` gruppen. Vi bruker gruppe-ID i stedet for et navn, siden vertssystemet kan bruke en annen ID for sitt `tty` gruppen.

mode=0620

Dette sikrer at alle `devpts` opprettede enhetsnoder har modus 0620 (bruker lesbar og skrivbar, gruppeskrivbar). Sammen med alternativet ovenfor, sikrer dette at `devpts` vil opprette enhetsnoder som oppfylle kravene til `grantpt()`, som betyr Glibc **pt_chown** hjelper binær (som ikke er installert som standard) ikke er nødvendig.

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

I andre vertssystemer `/dev/shm` er et monteringspunkt for en `tmpfs`. I så fall vil monteringen av `/dev` ovenfor bare opprette `/dev/shm` i `chroot` miljøet som en mappe. I denne situasjonen monterer vi eksplisitt en `tmpfs`:

```
if [ -h $LFS/dev/shm ]; then
    install -v -d -m 1777 $LFS$(realpath /dev/shm)
else
    mount -vt tmpfs -o nosuid,nodev tmpfs $LFS/dev/shm
fi
```

7.4. 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 \
    MAKEFLAGS="-j$(nproc)" \
    TESTSUITEFLAGS="-j$(nproc)" \
    /bin/bash --login
```

Hvis du ikke vil bruke alle tilgjengelige logiske kjerner, bytt ut $\$(nproc)$ med antall logiske kjerner du ønsker å bruke til å bygge pakker i dette kapittelet og de følgende kapitler. Testpakkene til noen pakker (spesielt Autoconf, Libtool, og Tar) i Kapittel 8 er ikke berørt av `MAKEFLAGS`, de bruker en `TESTSUITEFLAGS` miljøvariabel i stedet. Vi setter det her også for å kjøre disse testpakkene med flere kjerner.

`-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 setter `TERM` variabelen inne i chroot til samme verdi som utenfor chroot. Denne variabelen er nødvendig for at programmer som **vim** og **less** kan fungere skikkelig. Hvis andre variabler ønskes, som f.eks `CFLAGS` eller `CXXFLAGS`, er dette et bra sted å sette dem.

Fra dette tidspunktet er det ikke nødvendig å bruke `LFS` variabelen lenger fordi alt arbeid vil være begrenset til `LFS` filsystemet. **chroot** kommandoen kjører Bash skallet med rot (`/`) mappen satt til `$LFS`.

Merk at `/tools/bin` ikke er i `PATH`. Dette betyr at kryssverktøykjeden ikke lenger vil bli brukt.

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



Notat

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 kjernefilssystemene er montert som forklart i Seksjon 7.3.1, «Montering og fylling av `/dev`» og Seksjon 7.3.2, «Montering av det virtuelle kjernefilssystemer» 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.



Notat

Noen av mappene nevnt i denne delen kan allerede være opprettet tidligere med eksplisitte instruksjoner eller når du installerer noen pakker. De gjentas nedenfor for fullstendighet.

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

```
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/lib/locale
mkdir -pv /usr/local/{bin,lib,sbin}
mkdir -pv /usr/{,local}/share/{color,dict,doc,info,locale,man}
mkdir -pv /usr/{,local}/share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local}/share/man/man{1..8}
mkdir -pv /var/{cache,local,log,mail,opt,spool}
mkdir -pv /var/lib/{color,misc,locate}

ln -sfv /run /var/run
ln -sfv /run/lock /var/lock

install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
```

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

Den første modusendringen sikrer at ikke hvem som helst kan komme inn i `/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` mappene, men kan ikke fjerne 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`. I LFS, oppretter vi kun mapper som trengs. Du må imidlertid gjerne lage disse mappene.



Advarsel

FHS gir ikke mandat til at mappen `/usr/lib64` skal eksistere, og LFS redaktørene har bestemt seg for å ikke bruke den. For at instruksjonene i LFS og BLFS skal fungere riktig, er det viktig at denne mappen ikke eksisterer. Fra tid til annen bør du bekrefte at den ikke eksisterer, fordi det er enkelt å lage den utilsiktet, og dette vil sannsynligvis ødelegge systemet ditt.

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 tilfredsstillte verktøy som forventer å finne `/etc/mtab`, opprett følgende symbolske lenke:

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

Lag en grunnleggende `/etc/hosts` fil som blir 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
uidd:x:80:80:UUID Generation Daemon User:/dev/null:/usr/bin/false
nobody:x:65534:65534:Unprivileged User:/dev/null:/usr/bin/false
EOF
```

Selve passordet for `root` settes 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:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
input:x:24:
mail:x:34:
kvm:x:61:
uudd:x:80:
wheel:x:97:
users:x:999:
nogroup:x:65534:
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 <https://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. GID på 5 er mye brukt for `tty` gruppen, og tallet 5 er også brukt i `/etc/fstab` for `devpts` filsystemet. Alle andre gruppenavn og GID-er kan velges fritt av systemets administrator siden velskrevne programmer ikke er avhengige av GID-nummer, men bruk heller gruppens navn.

ID 65534 brukes av kjernen for NFS og separat brukernavneområder for ikke-tilordnede brukere (de finnes på NFS serveren eller den overordnede brukernavneområde, men «finnes ikke» på den lokale maskinen eller i det separate navnerommet). Vi tildeler `nobody` og `nogroup` for å unngå en ikke navngitt ID. Men andre distroer kan behandle denne IDen annerledes, så alle flyttbare programmer bør ikke være avhengig av denne tildelingen.

Noen tester i Kapittel 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 gruppenavnopløsning nå virke:

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

login, **agetty**, og **init** programmene (og andre) bruker en rekke loggfiler 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 pålogginger og utlogginger. `/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.



Notat

`/run/utmp` filen registrerer brukerne som for øyeblikket er pålogget. Denne filen er opprettet dynamisk i oppstartsskriptene.



Notat

`utmp`, `wtmp`, `btmp`, og `lastlog` filene bruker 32-biters heltall for tidsstempler og de vil være fundamentalt ødelagte etter år 2038. Mange pakker har sluttet å bruke dem og andre pakker kommer til å slutte å bruke dem. Det er sannsynligvis best å betrakte dem som avviklet.

7.7. Gettext-0.24

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: 1.3 SBU

Nødvendig diskplass: 349 MB

7.7.1. 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 Seksjon 8.33.2, «Innhold i Gettext»

7.8. Bison-3.8.2

Bison pakken inneholder en parsergenerator.

Omtrentlig byggetid: 0.2 SBU

Nødvendig diskplass: 58 MB

7.8.1. 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 Seksjon 8.34.2, «Innholdet i Bison»

7.9. Perl-5.40.1

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

Omtrentlig byggetid: 0.6 SBU

Nødvendig diskplass: 285 MB

7.9.1. Installasjon av Perl

Forbered Perl for kompilering:

```
sh Configure -des \
-D prefix=/usr \
-D vendorprefix=/usr \
-D useshrplib \
-D privlib=/usr/lib/perl5/5.40/core_perl \
-D archlib=/usr/lib/perl5/5.40/core_perl \
-D sitelib=/usr/lib/perl5/5.40/site_perl \
-D sitearch=/usr/lib/perl5/5.40/site_perl \
-D vendorlib=/usr/lib/perl5/5.40/vendor_perl \
-D vendorarch=/usr/lib/perl5/5.40/vendor_perl
```

Betydningen av 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-essensiell utdata.

-D vendorprefix=/usr
 Dette sikrer at **perl** vet hvordan å fortelle pakker hvor de skal installere perl modulene sine.

-D useshrplib
 Bygger `libperl` som trengs av noen perl moduler som et delt bibliotek, i stedet for et statisk bibliotek.

-D privlib,-D archlib,-D sitelib,...
 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 (5.40) hvilket tillater oppgradering av Perl til nyere Patch nivåer (Patchnivået er den siste punktseparerte delen i den fullstendige versjonenstrengen som 5.40.1) uten å installere alle modulene på nytt.

Kompiler pakken:

```
make
```

Installer pakken:

```
make install
```

Detaljer om denne pakken finner du i Seksjon 8.43.2, «Innhold i Perl»

7.10. Python-3.13.2

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

Omtrentlig byggetid: 0.5 SBU

Nødvendig diskplass: 634 MB

7.10.1. Installasjon av Python



Notat

Det er to pakkefiler som navnet begynner med «python» prefiks . Den å pakke ut er `Python-3.13.2.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
```



Notat

Noen Python 3 moduler kan ikke bygges nå på grunn av at avhengighetene ikke er installert ennå. For `ssl` modulen, en melding `Python requires a OpenSSL 1.1.1 or newer` sendes ut. 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 Kapittel 8.

Installer pakken:

```
make install
```

Detaljer om denne pakken finner du i Seksjon 8.51.2, «Innhold i Python 3»

7.11. Texinfo-7.2

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

Omtrentlig byggetid: 0.2 SBU

Nødvendig diskplass: 152 MB

7.11.1. Installasjon av Texinfo

Forbered Texinfo for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

```
make install
```

Detaljer om denne pakken finner du i Seksjon 8.72.2, «Innhold i Texinfo»

7.12. Util-linux-2.40.4

Util-linux pakken inneholder diverse hjelpeprogrammer.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 182 MB

7.12.1. 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 --libdir=/usr/lib \
            --runstatedir=/run \
            --disable-chfn-chsh \
            --disable-login \
            --disable-nologin \
            --disable-su \
            --disable-setpriv \
            --disable-runuser \
            --disable-pylibmount \
            --disable-static \
            --disable-liblastlog2 \
            --without-python \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.40.4
```

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 at `.so` målrettes direkte mot symbolkoblinger i den delte bibliotekfilen i samme mappe (`/usr/lib`).

```
--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 **uidd** og `libuidd` riktig.

Kompiler pakken:

```
make
```

Installer pakken:

```
make install
```

Detaljer om denne pakken finner du i Seksjon 8.79.2, «Innhold i Util-linux»

7.13. Rydde opp og lagre det midlertidige systemet

7.13.1. Rydde opp

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, på et moderne Linuxsystem, er libtool .la-filene bare nyttig for libltdl. Ingen biblioteker i LFS forventes å bli lastet av libltdl, og det er kjent at noen .la-filer kan forårsake at BLFS pakker feiler under byggingen. 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
```

7.13.2. 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 gjære om noe som har blitt bygget vellykket, det og lage en sikkerhetskopi av det nåværende LFS systemet kan vise seg å være nyttig.



Notat

Alle de resterende trinnene i denne delen er valgfrie. Likevel, så snart du begynner å installere pakker i Kapittel 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 burde plasseres innenfor \$LFS hierarkiet.

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

```
exit
```



Viktig

Alle følgende instruksjoner utføres av `root` på vertssystemet ditt. Vær ekstra forsiktig med kommandoene du skal kjøre ettersom feil her kan 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 Seksjon 2.6, «Stille inn \$LFS variabelen og Umask»

Får du lager en sikkerhetskopi, avmonter det virtuelle filsystemet:

```
mountpoint -q $LFS/dev/shm && umount $LFS/dev/shm
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 vertssystemets bruker `root` som vanligvis finnes på rotfilsystemet. Erstatt `$HOME` med en mappe etter eget valg hvis du ikke ønsker å ha sikkerhetskopien lagret i `root` sin hjemmemappe.

Opprett sikkerhetskopiarkivet ved å kjøre følgende kommando:



Notat

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

```
cd $LFS
tar -cJpf $HOME/lfs-temp-tools-12.3.tar.xz .
```



Notat

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

7.13.3. Gjenoppsett

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, gjenoppsett sikkerhetskopien ved å utføre følgende kommandoer:



Advarsel

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-12.3.tar.xz
```

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



Viktig

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 Seksjon 7.3, «Forberede det virtuelle kjernefilsystemet» og gå inn i chroot miljøet igjen (se Seksjon 7.4, «Gå inn i Chroot miljøet») før du fortsetter.

Del IV. Bygge LFS systemet

Kapittel 8. Installere grunnleggende systemprogramvare

8.1. Introduksjon

I dette kapitlet 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 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 kompileringsvanskeligheter 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 det er komplekse interaksjoner 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.

På den annen side holder vi optimaliseringene aktivert som standard konfigurasjon av pakkene. I tillegg aktiverer vi noen ganger eksplisitt en optimalisert konfigurasjon levert av en pakke, men ikke aktivert som standard. Pakkevedlikeholderne har allerede testet disse konfigurasjonene og anser dem som trygge, så det er ikke sannsynlig at de vil bryte byggingen. Vanligvis aktiverer standardkonfigurasjonen allerede `-O2` eller `-O3`, så det resulterende systemet vil fortsatt kjøre veldig raskt uten noen tilpasset optimalisering, og være stabil 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 som kreves under denne byggeprosessen. Etter installasjonsinstruksjonene, er det en liste over programmer og biblioteker (sammen med korte beskrivelser) som pakken installerer.



Notat

SBU verdiene og nødvendig diskplass inkluderer testpakkedata for alle gjeldende pakker i Kapittel 8. SBU verdier har blitt beregnet ved å bruke fire CPU kjerner (-j4) for alle operasjoner med mindre annet er spesifisert.

8.1.1. Om biblioteker

Generelt fraråder LFS redaktørene å bygge og installere statiske biblioteker. De fleste statiske biblioteker er gjort foreldet i et moderne Linuxsystem. I tillegg å koble et statisk bibliotek inn i et program kan være skadelig. Hvis en oppdatering til biblioteket er nødvendig for å fjerne et sikkerhetsproblem, må hvert program som bruker det statiske biblioteket kobles sammen med det nye biblioteket. Siden bruken av statiske biblioteker ikke alltid er åpenbart, de relevante programmene (og prosedyrene som trengs for å gjøre koblingen) er kanskje ikke engang kjent.

I prosedyrene i dette kapitlet 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 pakkebyggeprosessen.

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. En god pakkebehandler vil også håndtere konfigurasjonsfiler spesielt for å beholde brukerkonfigurasjonen når pakken installeres på nytt eller oppgraderes. Før du begynner å lure, 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 inkluderer:

- Å håndtere pakkehåndtering fjerner fokus fra målene til disse bøkene—å lære hvordan et Linux system er bygget.
- Det er flere løsninger for pakkehåndtering, som hver har dens styrker og ulemper. Inkludere en som tilfredsstillende 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 blir 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 oppgraderes ved siden av kjernen. Du må starte systemet på nytt for å bruke den oppgraderte kjernen.
- Hvis Glibc må oppgraderes til en nyere versjon, (f.eks. fra Glibc-2.36 til Glibc-2.41), noen ekstra trinn er nødvendig for å unngå å ødelegge systemet. Les Seksjon 8.5, «Glibc-2.41» for detaljer.
- 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 det forrige biblioteket med mindre alle de avhengige pakkene er rekompilert.
- Hvis en pakke er (direkte eller indirekte) er knyttet til både de gamle og nye navnene av et delt bibliotek (for eksempel pakken lenker til både `libfoo.so.2` og `libbar.so.1`, mens sistnevnte linker til `libfoo.so.3`), pakken kan fungere feil fordi de forskjellige revisjonene av det delte biblioteket presenterer uforenlige definisjoner for noen symbolnavn. Dette kan være forårsaket av rekompilering av noen, men ikke alle, pakkene knyttet til et gammelt delt bibliotek etter at pakken som gir det delte biblioteket er oppgradert. For å unngå problemet, må brukere gjenoppbygge hver pakke koblet til et delt bibliotek med en oppdatert revisjon (f.eks. `libfoo.so.2` til `libfoo.so.3`) så snart som mulig.
- Hvis en pakke som inneholder et delt bibliotek oppdateres, og navnet på biblioteket ikke endres, men versjonsnummeret til bibliotek **filen** 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). Ellers, et **ldconfig** kommando (påkalt av deg selv fra kommandolinjen, 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 oppstå hvis du må nedgradere en pakke, eller hvis forfatterne endrer versjonsskjema for bibliotekfiler.

- 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 '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` tjenesten, deretter logg ut, logg på igjen, og kjør kommandoen igjen 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 binærfilet 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 binærfilet og biblioteker. Dette betyr at du ikke vil bli plaget av dette problemet mesteparten av tiden. Imidlertid er installasjonsprosessen for noen pakker (spesielt SpiderMonkey 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. Får du ta en avgjørelse om en pakkeforvalter, gjør litt undersøkelser på de forskjellige teknikkene, spesielt ulempene ved den spesielle ordningen.

8.2.2.1. Alt er 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 mapper

Dette er en forenklet pakkehåndtering som ikke trenger noe ekstra pakke for å administrere installasjonene. Hver pakke er installert i en egen mappe. For eksempel pakke `foo-1.1` er installert i `/opt/foo-1.1` og en symbolkobling er laget fra `/opt/foo` til `/opt/foo-1.1`. Når en ny versjon `foo-1.2` kommer, blir den installert i `/opt/foo-1.2` og den forrige symbolkoblingen erstattes av en symbolkobling til den nye versjonen.

Miljøvariabler som f.eks `PATH`, `MANPATH`, `INFOPATH`, `PKG_CONFIG_PATH`, `CPPFLAGS`, `LDFLAGS`, og konfigurasjonsfilen `/etc/ld.so.conf` må kanskje utvides til inkludere de tilsvarende undermappene i `/opt/foo-x.y`.

Denne ordningen brukes av BLFS boken for å installere noen veldig store pakker for å gjøre det enklere å oppgradere dem. Hvis du installerer mer enn noen få pakker, blir denne ordningen uhåndterlig. Og noen pakker (for eksempel Linux API deklarasjoner og Glibc) fungerer kanskje ikke bra med denne ordningen. **Bruk aldri denne ordningen systemomfattende.**

8.2.2.3. Symlink Style Package Management

Dette er en variant av den tidligere pakkehåndteringsteknikken. Hver pakke er installert som i forrige skjema. Men i stedet for å lage symbolkoblingen via et generisk pakkenavn, er hver fil symlinket til `/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 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 gjøre at pakken ikke installeres 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` strategien for å forfalske installasjon av pakken. Denne 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 fordelene av å være enkel, har den to ulemper. Hvis filene under installasjonen er installert med et annet tidsstempel enn gjeldende tid, vil disse filene ikke spores av pakkebehandleren. Dessuten kan denne ordningen bare brukes når en 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` g 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 installasjonsskriptet.

8.2.2.6. Opprette pakkearkiver

I denne ordningen er pakkeinstallasjonen forfalsket til et separat tre som beskrevet i Symlink pakkebehandlingen. 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 denne tilnærmingen er RPM (som for øvrig kreves av *Linux Standard Base Specification*), `pkg-utils`, Debian sin `apt`, og Gentoo sin `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 <https://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. 900MB 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, /etc/ld.so.conf, /etc/sysconfig/rc.site, /etc/sysconfig/network, og /etc/sysconfig/ifconfig.eth0.

En tilpasset kjerne må kanskje bygges for det nye systemet avhengig av forskjeller i systemmaskinvare og den originale kjernekonfigurasjonen.



Notat

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 Seksjon 10.4, «Bruke GRUB til å sette opp oppstartsprosessen».

8.3. Man-pages-6.12

Man-pages pakken inneholder over 2400 mansider..

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 52 MB

8.3.1. Installasjon av Man-pages

Fjern to mansider for passordhashingsfunksjoner. Libxcrypt vil gi en bedre versjon av disse mansidene:

```
rm -v man3/crypt*
```

Installer Man-pages ved å kjøre:

```
make -R GIT=false prefix=/usr install
```

Betydningen av alternativene:

-R

Dette hindrer **make** fra å sette noen innebygde variabler. Byggesystemet for man-pages gjør det at det ikke fungerer bra med innebygde variabler, men foreløpig er det ingen måte å deaktivere dem unntatt å sende *-R* eksplisitt via kommandolinjen.

GIT=false

Dette hindrer byggesystemet fra å sende ut mange `git: command not found` advarsel linjer.

8.3.2. Innhold i Man-pages

Installerte filer: ulike mansider

Korte beskrivelser

`man pages` Beskriver C programmeringsspråksfunksjoner, viktig enhetsfiler og betydelige konfigurasjonsfiler

8.4. Iana-Etc-20250123

Iana-Etc pakken leverer data for nettverkstjenester og protokoller.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 4.8 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

<code>/etc/protocols</code>	Beskriver de ulike DARPA Internettprotokollene som er tilgjengelig fra TCP/IP undersystemet
<code>/etc/services</code>	Gir en tilordning mellom vennlige tekstnavn for internettjenester, og deres underliggende tildelte portnumre og protokolltyper

8.5. Glibc-2.41

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: 12 SBU
Nødvendig diskplass: 3.2 GB

8.5.1. Installation of 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-2.41-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=5.4   \
              --enable-stack-protector=strong \
              --disable-nscd        \
              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=5.4`

Dette alternativet forteller byggesystemet at denne glibc kan brukes med kjerner så gamle som 5.4 . 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 bufferoverflyt "buffer overflows", for eksempel stabel (stack) knusende angrep. Merk at Glibc alltid eksplisitt overstyrer standarden til GCC, så dette alternativet er fortsatt nødvendig selv om vi har allerede spesifisert `--enable-default-ssp` for GCC.

`--disable-nscd`

Ikke bygg navnetjenesten cache daemon som ikke er brukt lenger.

`libc_cv_slibdir=/usr/lib`

Denne variabelen setter riktig bibliotek for alle systemer. Vi ønsker ikke at lib64 skal brukes.

Kompilér pakken:

```
make
```

**Viktig**

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 6000 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.
- Noen tester, for eksempel *nss/tst-nss-files-hosts-multi* og *nptl/tst-thread-affinity** er kjent for å mislykkes på grunn av et tidsavbrudd (spesielt når systemet er relativt sakte og/eller kjører testpakken med flere parallelle make jobber). Disse testene kan identifiseres med:

```
grep "Timed out" $(find -name \*.out)
```

Det er mulig å kjøre en enkelt test på nytt med større tidsavbrudd med **TIMEOUTFACTOR=<factor> make test t=<test name>**. For eksempel, **TIMEOUTFACTOR=10 make test t=nss/tst-nss-files-hosts-multi** vil kjøre på nytt *nss/tst-nss-files-hosts-multi* med ti ganger det opprinnelige tidsavbruddet.

- I tillegg kan noen tester mislykkes med en relativt gammel CPU modell (For eksempel *elf/tst-cpu-features-cpuinfo*) eller vertskjerne versjon (for eksempel *stdlib/tst-arc4random-thread*).

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 Makefile for å hoppe over en utdatert tilregnlighetssjekk som mislykkes med en moderne Glibc-konfigurasjon:

```
sed '/test-installation/s@$(PERL)@echo not running@' -i ../Makefile
```



Viktig

Hvis du oppgraderer Glibc til en ny mindre versjon (for eksempel fra Glibc-2.36 to Glibc-2.41) på et kjørende LFS system, må du ta noen ekstra forholdsregler for å unngå å ødelegge systemet:

- Oppgradering av Glibc på et LFS system før 11.0 (eksklusivt) er ikke støttet. Bygg LFS på nytt hvis du kjører et så gammelt LFS system, men du trenger en nyere Glibc.
- Hvis du oppgraderer på et LFS system før 12.0 (eksklusivt), installer Libxcrypt med å følge Seksjon 8.27, «Libxcrypt-4.4.38» I tillegg til en normal Libxcrypt installasjon, **du MÅ følge notatet i Libxcrypt delen for å installere libxcrypt.so.1* (ved å erstatte libxcrypt.so.1 fra den tidligere Glibc installasjonen).**
- Hvis du oppgraderer på et LFS-system før 12.1 (eksklusivt), fjern **nscd** programmet:

```
rm -f /usr/sbin/nscd
```

- Oppgrader kjernen og start på nytt hvis den er eldre enn 5.4 (sjekk gjeldende versjon med **uname -r**) eller hvis du vil oppgradere den likevel, følg Seksjon 10.3, «Linux-6.13.4»
- Oppgrader kjerne API deklarasjonene hvis de er eldre enn 5.4 (sjekk gjeldende versjon med **cat /usr/include/linux/version.h**) eller hvis du vil oppgradere den likevel, følg Seksjon 5.4, «Linux-6.13.4 API Deklarasjoner» (men fjern `$LFS` fra **cp** kommandoen).
- Utfør en `DESTDIR` installasjon og oppgrader Glibc delte biblioteker på systemet ved hjelp av en enkelt **install** kommando:

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/*.so.* /usr/lib
```

Det er viktig å strengt følge disse trinnene ovenfor med mindre du forstår helt hva du gjør. **Ethvert uventet avvik kan gjøre systemet helt ubrukelig. DU ER ADVART.**

Fortsett deretter å kjøre **make install** kommandoen, **sed** kommandoen mot `/usr/bin/ldd`, og kommandoene som skal installere lokalitetene. Når de er ferdige, start systemet på nytt med en gang.

Når systemet har startet på nytt, hvis du kjører et LFS system før 12.0 (eksklusivt) der GCC ikke ble bygget med `--disable-fixincludes` alternativet, flytt to GCC deklarasjoner til en bedre plassering og fjern de foreldede «fixed» kopier av Glibc deklarasjonene:

```
DIR=$(dirname $(gcc -print-libgcc-file-name))
[ -e $DIR/include/limits.h ] || mv $DIR/include{-fixed,}/limits.h
[ -e $DIR/include/syslimits.h ] || mv $DIR/include{-fixed,}/syslimits.h
rm -rfv $(dirname $(gcc -print-libgcc-file-name))/include-fixed/*
```

Installer pakken:

```
make install
```

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

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

Installer deretter lokalitetene som kan få systemet til å svare i et annet språk. Ingen av disse stedene er påkrevd, men hvis noen av dem mangler, vil testpakkene til noen pakker hoppe over viktige testsaker.

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:

```
localedef -i C -f UTF-8 C.UTF-8
localedef -i cs_CZ -f UTF-8 cs_CZ.UTF-8
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i de_DE -f UTF-8 de_DE.UTF-8
localedef -i el_GR -f ISO-8859-7 el_GR
localedef -i en_GB -f ISO-8859-1 en_GB
localedef -i en_GB -f UTF-8 en_GB.UTF-8
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_ES -f ISO-8859-15 es_ES@euro
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i is_IS -f ISO-8859-1 is_IS
localedef -i is_IS -f UTF-8 is_IS.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i it_IT -f ISO-8859-15 it_IT@euro
localedef -i it_IT -f UTF-8 it_IT.UTF-8
localedef -i ja_JP -f EUC-JP ja_JP
localedef -i ja_JP -f SHIFT_JIS ja_JP.SJIS 2> /dev/null || true
localedef -i ja_JP -f UTF-8 ja_JP.UTF-8
localedef -i nl_NL@euro -f ISO-8859-15 nl_NL@euro
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i se_NO -f UTF-8 se_NO.UTF-8
localedef -i ta_IN -f UTF-8 ta_IN.UTF-8
localedef -i tr_TR -f UTF-8 tr_TR.UTF-8
localedef -i zh_CN -f GB18030 zh_CN.GB18030
localedef -i zh_HK -f BIG5-HKSCS zh_HK.BIG5-HKSCS
localedef -i zh_TW -f UTF-8 zh_TW.UTF-8
```

Installer i tillegg lokaliteten for ditt eget land, språk og tegnsett.

Alternativt kan du installere alle lokaliteter som er oppført i `glibc-2.41/localedata/SUPPORTED` filen (den inkluderer alle lokaliteter oppført ovenfor og mange flere) med en gang med denne tidkrevende kommandoen:

```
make localedata/install-locales
```

Bruk deretter **localedef** kommandoen for å lage og installere lokaliteter som ikke er oppført i `glibc-2.41/localedata/SUPPORTED` filen når du trenger dem. For eksempel er følgende to lokaliteter nødvendig for noen tester senere i dette kapittelet:

```
localedef -i C -f UTF-8 C.UTF-8
localedef -i ja_JP -f SHIFT_JIS ja_JP.SJIS 2> /dev/null || true
```



Notat

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. Konfigurere 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. Legge til tidssonedata

Installer og sett opp tidssonedataene med følgende:

```
tar -xf ../../tzdata2025a.tar.gz

ZONEINFO=/usr/share/zoneinfo
mkdir -pv $ZONEINFO/{posix,right}

for tz in etcetera southamerica northamerica europe africa antarctica \
        asia australasia backward; do
    zic -L /dev/null -d $ZONEINFO      ${tz}
    zic -L /dev/null -d $ZONEINFO/posix ${tz}
    zic -L leapseconds -d $ZONEINFO/right ${tz}
done

cp -v zone.tab zone1970.tab iso3166.tab $ZONEINFO
zic -d $ZONEINFO -p Europe/Oslo
unset ZONEINFO tz
```

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 spare 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. *Europe/Oslo*).

8.5.2.3. Konfigurere den dynamiske lasteren

Som standard den dynamiske lasteren (`/lib/ld-linux.so.2`) søker gjennom `/usr/lib` for dynamiske biblioteker som trengs av programmer når de kjøres. Imidlertid, hvis det er biblioteker i andre kataloger enn `/usr/lib`, må disse legges til `/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 (symlenke til `ld-linux-x86-64.so.2` or `ld-linux.so.2`), locale, localedef, makedb, mtrace, pccprofiledump, 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`, `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`, 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 lokalitetsspesifikasjoner
makedb	Oppretter en enkel database fra tekst inndata
mtrace	Leser og tolker en minnesporingsfil og viser et sammendrag i menneskelestbart format
pcprofiledump	Dumper informasjon generert av PC profiling
pldd	Viser dynamiske delte objekter som brukes av kjørende prosesser
sln	En statisk koblet ln program
sotrust	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
<code>ld-*.so</code>	Hjelpeprogrammet for kjørbare delte biblioteker
<code>libBrokenLocale</code>	Brukes internt av Glibc som et grovt hack for å få ødelagte programmer (f.eks. noen Motif-applikasjoner) kjørende. Se kommentarer i <code>glibc-2.41/locale/broken_cur_max.c</code> for mer informasjon
<code>libanl</code>	Dummy bibliotek som ikke inneholder noen funksjoner. Tidligere var det asynkront navneoppslagsbibliotek, hvor funksjoner nå er i <code>libc</code>
<code>libc</code>	C hovedbiblioteket
<code>libc_malloc_debug</code>	Slår på minneallokeringskontroll når den er forhåndslestet
<code>libdl</code>	Dummy bibliotek som ikke inneholder noen funksjoner. Tidligere var den dynamisk koblingsgrensesnittbibliotek, funksjonene er nå i <code>libc</code>
<code>libg</code>	Dummy bibliotek som ikke inneholder noen funksjoner. Tidligere var det et kjøretidsbibliotek for g++
<code>libm</code>	Det matematiske biblioteket
<code>libmvec</code>	Matematisk vektorbibliotek, koblet inn etter behov når <code>libm</code> blir brukt
<code>libmcheck</code>	Slår på minneallokeringskontroll når den er koblet til
<code>libmemusage</code>	Brukt av memusage for å hjelpe til med å samle inn informasjon om minnebruken til et program

<code>libnsl</code>	Nettverkstjenestebiblioteket, nå avviklet
<code>libnss_*</code>	Navnetjenestebrytermodulene, som inneholder funksjoner for å løse vertsnavn, brukernavn, gruppenavn, aliaser, tjenester, protokoller osv. Lastet av <code>libc</code> ifølge konfigurasjon i <code>/etc/nsswitch.conf</code>
<code>libpcprofile</code>	Kan forhåndslastes til PC profile en kjørbare fil
<code>libpthread</code>	Dummy bibliotek som ikke inneholder noen funksjoner. Tidligere inneholdt den funksjoner som gir de fleste grensesnittene som er spesifisert av POSIX.1c Threads Extensions og semafor grensesnittene spesifisert av POSIX.1b Real-time Extensions, nå er funksjonene i <code>libc</code>
<code>libresolv</code>	Inneholder funksjoner for å lage, sende og tolke pakker til domenenavnservere
<code>librt</code>	Inneholder funksjoner som gir de fleste grensesnittene som er spesifisert av POSIX.1b Real-time Extensions
<code>libthread_db</code>	Inneholder funksjoner som er nyttige for å bygge feilsøkere for flertrådede programmer
<code>libutil</code>	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 <code>libc</code>

8.6. Zlib-1.3.1

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

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 6.4 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: less than 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 mandsidene 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 bzmored), og bzmored

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
<code>libbz2</code>	Biblioteket implementerer tapsfri, blokksorterende datakomprimering ved å bruke Burrows-Wheeler-algoritmen

8.8. Xz-5.6.4

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

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 21 MB

8.8.1. Installasjon av Xz

Forbered Xz for kompilering med:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.6.4
```

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), lzdifff (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-5.6.4

Korte beskrivelser

lzcat	Dekomprimerer til standard utgang
lzcmp	Kjører cmp på LZMA komprimerte filer
lzdifff	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 dekodeer 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
<code>liblzma</code>	Bibliotek som implementerer tapsfri, blokksorterende data komprimering ved å bruke Lempel-Ziv-Markov kjedevalgoritmen

8.9. Lz4-1.10.0

Lz4 er en tapsfri komprimeringsalgoritme som gir kompresjonshastighet mer enn 500 MB/s per kjerne. Den har en ekstremt rask dekode, med hastighet i flere GB/s per kjerne. Lz4 kan fungere med Zstandard for å tillate begge algoritmene for å komprimere data raskere.

Omtrentlig byggetid: 0.1 SBU

Nødvendig diskplass: 4.2 MB

8.9.1. Installasjon av Lz4

Kompiler pakken:

```
make BUILD_STATIC=no PREFIX=/usr
```

For å teste resultatene, utsted:

```
make -j1 check
```

Installer pakken:

```
make BUILD_STATIC=no PREFIX=/usr install
```

8.9.2. Innhold i Lz4

Installerte programmer: lz4, lz4c (lenke til lz4), lz4cat (lenke til lz4), og unlz4 (lenke til lz4)

Installert bibliotek: liblz4.so

Korte beskrivelser

lz4	Komprimerer eller dekomprimerer filer ved hjelp av LZ4 formatet
lz4c	Komprimerer filer ved hjelp av LZ4 formatet
lz4cat	Viser innholdet i en fil komprimert ved hjelp av LZ4 formatet
unlz4	Dekomprimerer filer ved hjelp av LZ4 formatet
<code>liblz4</code>	Biblioteket som implementerer tapsfrie data komprimering ved hjelp av LZ4 algoritmen

8.10. Zstd-1.5.7

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 dekode.

Omtrentlig byggetid: 0.4 SBU

Nødvendig diskplass: 85 MB

8.10.1. Installasjon av Zstd

Kompilér pakken:

```
make prefix=/usr
```



Notat

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
```

8.10.2. 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 datakomprimering ved å bruke ZSTD algoritmen

8.11. File-5.46

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

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 19 MB

8.11.1. Installasjon av File

Forbered File for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.11.2. 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

8.12. Readline-8.2.13

Readline pakken er et sett med biblioteker som tilbyr redigerings- og historikkfunksjoner på kommandolinjen.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 16 MB

8.12.1. 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 unngås ved å utstede følgende to seds:

```
sed -i '/MV.*old/d' Makefile.in
sed -i '{OLDSUFF}/c:' support/shlib-install
```

Forhindre hardkoding biblioteksøkestier (rpath) inn i de delte bibliotekene. Denne pakken trenger ikke rpath for en installasjon på standardplasseringen, og rpath kan noen ganger forårsake uønskede effekter eller til og med sikkerhetsproblemer:

```
sed -i 's/-Wl,-rpath,[^ ]*//' support/shobj-conf
```

Forbered Readline for kompilering:

```
./configure --prefix=/usr \
            --disable-static \
            --with-curses \
            --docdir=/usr/share/doc/readline-8.2.13
```

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. For detaljer se «Shared Libraries» seksjonen i pannen sin `README` fil.

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make install
```

Hvis ønskelig, installer dokumentasjonen:

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.2.13
```

8.12.2. Innhold i Readline

Installerte biblioteker: libhistory.so og libreadline.so
Installerte mapper: /usr/include/readline og /usr/share/doc/readline-8.2.13

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 i en interaktiv økt av et program

8.13. M4-1.4.19

M4 pakken inneholder en makroprosessor.

Omtrentlig byggetid: 0.3 SBU

Nødvendig diskplass: 49 MB

8.13.1. Installasjon av M4

Forbered M4 for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.13.2. 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, utføre heltallsaritmetikk, manipulere tekst, rekursjon osv. **m4** programmet kan brukes enten som en grenseflate til en kompilator eller som en makroprosessor på egen hånd

8.14. Bc-7.0.3

Bc pakken inneholder et vilkårlig behandlingsspråk for numerisk presisjon.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 7.8 MB

8.14.1. Installasjon av Bc

Forbered Bc for kompilering:

```
CC=gcc ./configure --prefix=/usr -G -O3 -r
```

Betydningen av konfigureringsalternativene:

CC=gcc

Denne parameteren spesifiserer kompilatoren som skal brukes.

-G

Utelat deler av testpakken som ikke vil fungere før bc programmet er installert.

-O3

Spesifiser optimaliseringen som skal brukes.

-r

Aktiver bruk av Readline for å forbedre linjeredigeringsfunksjonen til bc.

Kompiler pakken:

```
make
```

For å teste bc, kjør

```
make test
```

Installer pakken:

```
make install
```

8.14.2. Innholdet i Bc

Installerte programmer: bc og dc

Korte beskrivelser

bc En kommandolinjekalkulator

dc En reverse-polish kommandolinjekalkulator

8.15. Flex-2.6.4

Flex pakken inneholder et verktøy for å generere programmerer som gjenkjenner mønstre i tekst.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 33 MB

8.15.1. 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, 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, og opprett også mansiden til **lex** som en symlenke:

```
ln -sv flex /usr/bin/lex  
ln -sv flex.1 /usr/share/man/man1/lex.1
```

8.15.2. 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

8.16. Tcl-8.6.16

Tcl pakken inneholder Tool Command Language, et robust skriptspråk for generelt bruk. Expect pakken er skrevet i Tcl (uttales "tickle").

Omtrentlig byggetid: 3.1 SBU

Nødvendig diskplass: 91 MB

8.16.1. 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.

Forbered Tcl for kompilering:

```
SRCDIR=$(pwd)
cd unix
./configure --prefix=/usr \
            --mandir=/usr/share/man \
            --disable-rpath
```

Betydningen av de nye konfigureringsparametrene:

--disable-rpath

Denne parameteren forhindrer hardkoding av biblioteksøkebaner (rpath) inn i de binære kjørbare filene og delte bibliotekene. Denne pakken trenger ikke rpath for en installasjon i standard plassering, og rpath kan noen ganger forårsake uønskede effekter eller til og med sikkerhetsproblemer.

Bygg pakken:

```
make

sed -e "s|${SRCDIR}/unix|/usr/lib|" \
    -e "s|${SRCDIR}|/usr/include|" \
    -i tclConfig.sh

sed -e "s|${SRCDIR}/unix/pkgs/tdbc1.1.10|/usr/lib/tdbc1.1.10|" \
    -e "s|${SRCDIR}/pkgs/tdbc1.1.10/generic|/usr/include|" \
    -e "s|${SRCDIR}/pkgs/tdbc1.1.10/library|/usr/lib/tcl8.6|" \
    -e "s|${SRCDIR}/pkgs/tdbc1.1.10|/usr/include|" \
    -i pkgs/tdbc1.1.10/tdbcConfig.sh

sed -e "s|${SRCDIR}/unix/pkgs/itcl4.3.2|/usr/lib/itcl4.3.2|" \
    -e "s|${SRCDIR}/pkgs/itcl4.3.2/generic|/usr/include|" \
    -e "s|${SRCDIR}/pkgs/itcl4.3.2|/usr/include|" \
    -i pkgs/itcl4.3.2/itclConfig.sh

unset SRCDIR
```

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, kjør:

```
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 manside som er i konflikt med en manside for Perl:

```
mv /usr/share/man/man3/{Thread,Tcl_Thread}.3
```

Eventuelt kan du installere dokumentasjonen ved å utstede følgende kommandoer:

```
cd ..
tar -xf ../tcl8.6.16-html.tar.gz --strip-components=1
mkdir -v -p /usr/share/doc/tcl-8.6.16
cp -v -r ./html/* /usr/share/doc/tcl-8.6.16
```

8.16.2. Innhold i Tcl

Installerte programmer: tclsh (lenker til 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

8.17. 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

8.17.1. Installasjon av Expect

Expect forventer at PTY-er skal fungere. Kontroller at PTY-ene fungerer riktig inne i chroot miljøet ved å utføre en enkel test:

```
python3 -c 'from pty import spawn; spawn(["echo", "ok"])'
```

Denne kommandoen skal sende ut `ok`. Hvis i stedet utdataen inkluderer `OSError: out of pty devices`, da er ikke miljøet satt opp for ordentlig PTY operasjon. Du må gå ut av chroot miljøet, les Seksjon 7.3, «Forberede det virtuelle kjernefilssystemet» igjen, og sørg for at `devpts` filsystemet (og andre virtuelle kjernefilssystemer) er montert på riktig måte. Gå deretter inn i chroot miljøet igjen ved å følge Seksjon 7.4, «Gå inn i Chroot miljøet». Dette problemet må løses før du fortsetter, ellers vil testpakkene som krever Expect (for eksempel testpakkene til Bash, Binutils, GCC, GDBM, og selvfølgelig Expect seg selv) vil mislykkes katastrofalt, og andre subtile brudd kan også skje.

Gjør nå noen endringer for å tillate pakken med `gcc-14.1` eller nyere:

```
patch -Np1 -i ../expect-5.45.4-gcc14-1.patch
```

Forbered Expect for kompilering:

```
./configure --prefix=/usr \
            --with-tcl=/usr/lib \
            --enable-shared \
            --disable-rpath \
            --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
```

8.17.2. Innhold i Expect

Installert program: expect
Installert bibliotek: libexpect5.45.4.so

Korte beskrivelser

expect	Kommuniserer med andre interaktive programmer iht. til et skript
<code>libexpect-5.45.4.so</code>	Inneholder funksjoner som gjør at Expect kan brukes som en Tcl utvidelse eller brukes direkte fra C eller C++ (uten Tcl)

8.18. 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: less than 0.1 SBU
Nødvendig diskplass: 6.9 MB

8.18.1. 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
```

For å teste resultatene, kjør:

```
make check
```

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
```

8.18.2. 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

8.19. Pkgconf-2.3.0

Pkgconf pakken er en etterfølger til pkg-config og inneholder et verktøy for å sende inkluderingsbanen og/eller bibliotekstier for å bygge verktøy under konfigurerings- og makefasene av pakkeinstallasjoner.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 4.7 MB

8.19.1. Installasjon av Pkgconf

Forbered Pkgconf for kompilering:

```
./configure --prefix=/usr          \
            --disable-static       \
            --docdir=/usr/share/doc/pkgconf-2.3.0
```

Forbered Pkgconf for kompilering:

```
make
```

Installer pakken:

```
make install
```

For å opprettholde kompatibilitet med den originale Pkg-config oppretter du to symbolkoblinger:

```
ln -sv pkgconf /usr/bin/pkg-config
ln -sv pkgconf.1 /usr/share/man/man1/pkg-config.1
```

8.19.2. Innhold i Pkgconf

Installerte programmer: pkgconf, pkg-config (linker til pkgconf), og bomtool
Installert bibliotek: libpkgconf.so
Installert mappe: /usr/share/doc/pkgconf-2.3.0

Korte beskrivelser

pkgconf	Returnerer metainformasjon for det angitte biblioteket eller pakkenn
bomtool	Genererer en programvareliste fra pkg-config .pc filer
libpkgconf	Inneholder det meste av pkgconf sin funksjonalitet, mens det tillater andre verktøy som IDEer og kompilatorer å bruke rammeverket

8.20. Binutils-2.44

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

Omtrentlig byggetid: 1.6 SBU
Nødvendig diskplass: 819 MB

8.20.1. Installasjon av Binutils

Binutils dokumentasjonen anbefaler å bygge Binutils i en dedikert byggemappe:

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

Forbered Binutils for kompilering:

```
../configure --prefix=/usr \
              --sysconfdir=/etc \
              --enable-ld=default \
              --enable-plugins \
              --enable-shared \
              --disable-werror \
              --enable-64-bit-bfd \
              --enable-new-dtags \
              --with-system-zlib \
              --enable-default-hash-style=gnu
```

Betydningen av nye konfigureringsparametrene:

--enable-ld=default

Bygg den originale bfd linker og installer den som både ld (som er standard linker) og ld.bfd.

--enable-plugins

Aktiverer støtte for programtillegg for linker.

--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 verktøymappen (mappen der de kjørbare filene til slutt vil 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).



Viktig

Testpakken for Binutils i denne delen anses som kritisk. Ikke hopp over det under noen omstendigheter.

Test resultatene:

```
make -k check
```

For en liste over mislykkede tester, kjør:

```
grep '^FAIL:' $(find -name '*.log')
```

Installer pakken:

```
make tooldir=/usr install
```

Fjern ubrukelige statiske biblioteker og andre filer:

```
rm -rfv /usr/lib/lib{bfd,ctf,ctf-nobfd,gprofng,opcodes,sframe}.a \
/usr/share/doc/gprofng/
```

8.20.2. Innhold i Binutils

Installerte programmer: addr2line, ar, as, c++filt, dwp, elfedit, gprof, gprofng, ld, ld.bfd, nm, objcopy, objdump, ranlib, readelf, size, strings, og strip

Installerte biblioteker: libbfd.so, libctf.so, libctf-nobfd.so, libgprofng.so, libopcodes.so, og libsframe.so

Installert mappe: /usr/lib/ldscripts

Korte beskrivelser

addr2line Oversetter programadresser til filnavn og linjenumre; gitt en adresse og navnet på en kjørbare fil, bruker den feilsøking sin informasjon i den kjørbare filen for å bestemme hvilken kildefil og linjenummer 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 linkerens til å dekode C++ og Java symboler og hindre overbelastede funksjoner å krasje

dwp DWARF pakkeverktøyet

elfedit Oppdaterer ELF deklarasjonen til ELF filer

gprof Viser profildata for kallgrafene

gprofng Samler og analyser ytelsesdata

ld En linker som kombinerer en rekke objekt og arkivfiler inn i en enkelt fil, flytter dataene deres og rydder opp i symbolreferanser

ld.bfd Hardlenke til **ld**

nm Lister symbolene som forekommer i en gitt objektfil

objcopy Oversetter en 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 Fjerner symboler fra objektfiler

libbfd Biblioteket med binære filbeskrivelser

libctf Compat ANSI-C Type Format støttebibliotek for feilsøking

<code>libctf-nobfd</code>	En libctf variant som ikke bruker libbfd funksjonalitet
<code>libgprofng</code>	Et bibliotek som inneholder de fleste rutiner som brukes av gprofng
<code>libopcodes</code>	Et bibliotek for å håndtere opkoder—«leselig tekst» versjoner av instruksjoner for prosessoren; den brukes til å bygge verktøy som objdump
<code>libsframe</code>	Et bibliotek for å støtte tilbakesporing på nettet ved hjelp av en enkel avvikling

8.21. GMP-6.3.0

GMP pakken inneholder matematikkbiblioteker. Disse har nyttige funksjoner for vilkårlig presisjonsaritmetikk.

Omtrentlig byggetid: 0.3 SBU

Nødvendig diskplass: 54 MB

8.21.1. Installasjon av GMP



Notat

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 konfigureringskriptet forsøke å konfigurere for 64-biter og mislykkes. Unngå dette ved å påkalle `configure` kommandoen nedenfor med

```
ABI=32 ./configure ...
```



Notat

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 legge til `--host=none-linux-gnu` alternativet til **configure** kommandoen.

Forbered GMP for kompilering:

```
./configure --prefix=/usr \
            --enable-cxx \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.3.0
```

Betydningen av de nye konfigureringsalternativene:

`--enable-cxx`

Denne parameteren aktiverer C++ støtte

`--docdir=/usr/share/doc/gmp-6.3.0`

Denne variabelen spesifiserer riktig sted for dokumentasjon.

Kompiler pakken og generer HTML dokumentasjonen:

```
make
make html
```



Viktig

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
```



Obs

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 `--host=none-linux-gnu` og gjenoppbygges.

Sørg for at minst 199 tester 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
```

8.21.2. Innhold i GMP

Installerte biblioteker: libgmp.so og libgmpxx.so

Installert mappe: /usr/share/doc/gmp-6.3.0

Korte beskrivelser

libgmp Inneholder matematiske presisjonsfunksjoner

libgmpxx Inneholder C++ matematiske presisjonsfunksjoner

8.22. MPFR-4.2.1

MPFR pakken inneholder matematiske funksjoner med flere presisjoner.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 43 MB

8.22.1. Installasjon av MPFR

Forbered MPFR for kompilering:

```
./configure --prefix=/usr      \  
            --disable-static   \  
            --enable-thread-safe \  
            --docdir=/usr/share/doc/mpfr-4.2.1
```

Kompiler pakken og generer HTML dokumentasjonen:

```
make  
make html
```



Viktig

Testpakken for MPFR i denne delen anses som kritisk. Ikke hopp over den under noen omstendigheter.

Test resultatene og sørg for at alle 198 testene består:

```
make check
```

Installer pakken og dens dokumentasjon:

```
make install  
make install-html
```

8.22.2. Innhold i MPFR

Installerte biblioteker: libmpfr.so
Installert mappe: /usr/share/doc/mpfr-4.2.1

Korte beskrivelser

`libmpfr` Inneholder matematiske funksjoner med flere presisjoner

8.23. MPC-1.3.1

MPC pakken inneholder et bibliotek for aritmetikk av komplekse tall med vilkårlig høy presisjon og korrekt avrunding av resultatet.

Omtrentlig byggetid: 0.1 SBU

Nødvendig diskplass: 22 MB

8.23.1. Installasjon av MPC

Forbered MPC for kompilering:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.3.1
```

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
```

8.23.2. Innhold i MPC

Installerte biblioteker: libmpc.so

Installert mappe: /usr/share/doc/mpc-1.3.1

Korte beskrivelser

`libmpc` Inneholder komplekse matematiske funksjoner

8.24. Attr-2.5.2

Attr pakken inneholder verktøy for å administrere utvidede attributter på filsystemobjekter.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 4.1 MB

8.24.1. Installasjon av Attr

Forbered Attr for kompilering:

```
./configure --prefix=/usr      \  
            --disable-static  \  
            --sysconfdir=/etc  \  
            --docdir=/usr/share/doc/attr-2.5.2
```

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
```

8.24.2. Innhold i Attr

Installerte programmer: attr, getfattr, og setfattr
Installert bibliotek: libattr.so
Installerte mapper: /usr/include/attr og /usr/share/doc/attr-2.5.2

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

8.25. Acl-2.3.2

Acl pakken inneholder verktøy for å administrere tilgangskontrollister, som brukes til å definere mer finmaskede skjønnsmessige tilgangsrettigheter for filer og mapper.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 6.5 MB

8.25.1. Installasjon av Acl

Forbered Acl for kompilering:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/acl-2.3.2
```

Kompiler pakken:

```
make
```

Acl testene må kjøres på et filsystem som støtter tilgangskontroller. For å teste resultatene, utsted:

```
make check
```

En test navngitt `test/cp.test` er kjent for å feile fordi Coreutils ikke er bygget med Acl støtte enda.

Installer pakken:

```
make install
```

8.25.2. Innhold i Acl

Installerte programmer: chacl, getfacl, og setfacl
Installert bibliotek: libacl.so
Installerte mapper: /usr/include/acl og /usr/share/doc/acl-2.3.2

Korte beskrivelser

chacl Endrer tilgangskontrollisten til en fil eller mappe
getfacl Henter filtilgangskontrollister
setfacl Angir filtilgangskontrollister
libacl Inneholder bibliotekfunksjonene for å manipulere tilgangskontrollister

8.26. Libcap-2.73

Libcap pakken implementerer brukergrensesnittet 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: less than 0.1 SBU

Nødvendig diskplass: 3.0 MB

8.26.1. 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 bibliotekmappen til `/usr/lib` i stedet for `/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
```

8.26.2. 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 funksjonsstøtte
getcap	Undersøker filfunksjoner
getpcaps	Viser egenskapene til de forespurte prosessen(e)
setcap	Angir filfunksjoner
<code>libcap</code>	Inneholder bibliotekfunksjonene for å manipulere POSIX 1003.1e funksjoner
<code>libpsx</code>	Inneholder funksjoner for å støtte POSIX semantikk for syscalls knyttet til pthread biblioteket

8.27. Libxcrypt-4.4.38

Libxcrypt pakken inneholder et moderne bibliotek for enveis hashing av passord.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 12 MB

8.27.1. Installasjon av Libxcrypt

Forbered Libxcrypt for kompilering:

```
./configure --prefix=/usr \
            --enable-hashes=strong,glibc \
            --enable-obsolete-api=no \
            --disable-static \
            --disable-failure-tokens
```

Betydningen av de nye konfigureringsalternativene:

--enable-hashes=strong,glibc

Bygg sterke hash-algoritmer som anbefales for sikkerhetsbruk og hash-algortimene levert av tradisjonelle Glibc libcrypt for kompatibilitet.

--enable-obsolete-api=no

Deaktiver foreldede API-funksjoner. De trengs ikke for et moderne Linuxsystem bygget fra kilden.

--disable-failure-tokens

Deaktiver funksjonen for feiltoken. Det trengs for kompatibilitet med de tradisjonelle hash-bibliotekene til noen plattformer, men et Linuxsystem basert på Glibc trenger ikke den.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```



Notat

Instruksjonene ovenfor har deaktivert foreldede API-funksjoner siden ingen pakke installert ved å kompilere fra kilder ville lenke mot dem under kjøring. Imidlertid er de eneste kjente bare binære applikasjonene som koblingen mot disse funksjonene krever ABI versjon 1. Hvis du må ha slike funksjoner på grunn av en eller annen binær applikasjon eller for å være kompatibel med LSB, bygg pakken på nytt med følgende kommandoer:

```
make distclean
./configure --prefix=/usr \
            --enable-hashes=strong,glibc \
            --enable-obsolete-api=glibc \
            --disable-static \
            --disable-failure-tokens
make
cp -av --remove-destination .libs/libcrypt.so.1* /usr/lib
```

8.27.2. Innhold i Libxcrypt

Installerte biblioteker: libcrypt.so

Korte beskrivelser

`libcrypt` Inneholder funksjoner for å hashe passord

8.28. Shadow-4.17.3

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

Omtrentlig byggetid: 0.1 SBU

Nødvendig diskplass: 114 MB

8.28.1. Installasjon av Shadow



Viktig

Hvis du har installert Linux-PAM, bør du følge *BLFS instruksjon* i stedet for å bygge fra denne siden (eller, gjenoppbygge eller oppgradere) shadow.



Notat

Hvis du ønsker å håndheve bruken av sterke passord, *installer og konfigurere Linux-PAM* først. Deretter *installer og konfigurere shadow med PAM støtte*. Til slutt *installer libpwquality og konfigurere PAM til å bruke den*.

Deaktiver installasjonen av **groups** programmet og mandsidene, ettersom Coreutils gir en bedre versjon. Også forhindre installasjon av mandsider som allerede var installert i Seksjon 8.3, «Man-pages-6.12»:

```
sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i 's/groups\.1 / /' {} \;
find man -name Makefile.in -exec sed -i 's/getspnam\.3 / /' {} \;
find man -name Makefile.in -exec sed -i 's/passwd\.5 / /' {} \;
```

I stedet for å bruke standard *crypt* metoden, bruk den mye sikrere *YESCRYPT* metode for passordkryptering, som også tillater passord lengre enn 8 tegn. It is also necessary to change det foreldede `/var/spool/mail` plasseringen for brukerpostbokser som Shadow bruker som standard til `/var/mail` stedet som brukes for øyeblikket. Og, fjern `/bin` og `/sbin` fra `PATH`, siden de ganske enkelt er symbolske lenker til motparten i `/usr`.



Advarsel

Å inkludere `/bin` og/eller `/sbin` i `PATH` variabelen kan føre til at enkelte BLFS pakker mislykkes å bygge, så ikke gjør det i `.bashrc` filen eller andre steder.

```
sed -e 's:#ENCRYPT_METHOD DES:ENCRYPT_METHOD YESCRYPT:' \
-e 's:/var/spool/mail:/var/mail:' \
-e '/PATH={s@/sbin:@;s@/bin:@}' \
-i etc/login.defs
```

Forbered Shadow for kompilering:

```
touch /usr/bin/passwd
./configure --sysconfdir=/etc \
--disable-static \
--with-{b,yes}crypt \
--without-libbsd \
--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; hvis den ikke eksisterer allerede, vil installasjonsskriptet lage den på feil sted.

```
--with-{b,yes}crypt
```

Skallet utvider dette til to brytere, `--with-bcrypt` og `--with-yescrypt`. De lar shadow bruke Bcrypt- og Yescrypt-algortimene implementert av Libxcrypt for hashing av passord. Disse algoritmene er sikrere (spesielt mye mer motstandsdyktig mot GPU-baserte angrep) enn den tradisjonelle SHA algoritmer.

```
--with-group-name-max-length=32
```

Det lengste tillatte brukernavnet er 32 tegn. Gjør det maksimale lengden på et gruppenavn til det samme.

```
--without-libbsd
```

Ikke bruk readpassphrase funksjonen fra libbsd som ikke er i LFS. Bruk den interne kopien i stedet.

Kompiler pakken:

```
make
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make exec_prefix=/usr install
make -C man install-man
```

8.28.2. 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 *passordskygging* betyr, se `doc/HOWTO` filen i den utpakkede kildetreet. Hvis du bruker Shadowstøtte, husk at programmer som trenger å bekrefte passord (skjermbehandlere, FTP-programmer, pop3-nisser, etc.) må være Shadowkompatibel. 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 trenger litt forklaring. Først standard handling for **useradd** verktøyet er å lage brukeren og en gruppe med samme navn som brukeren. Som standard begynner brukerID (UID) og gruppeID 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 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` parameteren ovenfor. Du kan endre den til alt du ønsker. Merk at **useradd** vil aldri gjenbruke en UID eller GID. Hvis nummeret som er identifisert i denne parameteren brukes, vil den 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, vil en feilmelding bli generert—`useradd: unknown GID 999`, selv om kontoen er riktig opprettet. Det er derfor vi har opprettet gruppen `users` med denne gruppe-IDen i Seksjon 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 ikke vil opprette disse filene, gi følgende kommando:

```
sed -i '/MAIL/s/yes/no/' /etc/default/useradd
```

8.28.3. Sette root passordet

Velg et passord for brukeren *root* og sett det ved å kjøre:

```
passwd root
```

8.28.4. Innhold i Shadow

Installerte programmer: chage, chfn, chgpaswd, chpaswd, chsh, expiry, faillog, getsubids, gpaswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, login, logoutd, newgidmap, newgrp, newuidmap, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (lenker til newgrp), su, useradd, userdel, usermod, vigr (lenker til vipw), og vipw

Installerte mapper: /etc/default og /usr/include/shadow

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
chgpaswd	Brukes til å oppdatere gruppepassord i skriptmodus
chpaswd	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
gpaswd	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 <code>/etc/group</code> og <code>/etc/gshadow</code>
grpconv	Oppretter eller oppdaterer skyggegruppefilen fra den normale gruppefilen
grpunconv	Oppdaterer <code>/etc/group</code> fra <code>/etc/gshadow</code> og sletter deretter sistnevnte
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 som sier at en konto ikke er tilgjengelig; den er designet til å brukes som standard skall for deaktiverte kontoer
passwd	Brukes til å endre passordet for en bruker- eller gruppekonto
pwck	Verifiserer integriteten til passordfilene <code>/etc/passwd</code> og <code>/etc/shadow</code>
pwconv	Oppretter eller oppdaterer skyggepassordfilen fra den normale passordfilen
pwunconv	Oppdaterer <code>/etc/passwd</code> fra <code>/etc/shadow</code> 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 <code>/etc/group</code> eller <code>/etc/gshadow</code> filene
vipw	Redigerer <code>/etc/passwd</code> eller <code>/etc/shadow</code> filene
<code>libsubid</code>	bibliotek for å prosessere underordnede id-områder for brukere

8.29. GCC-14.2.0

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

Omtrentlig byggetid: 46 SBU (with tests)

Nødvendig diskplass: 6.3 GB

8.29.1. Installasjon av GCC

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
```

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++ \
             --enable-default-pie    \
             --enable-default-ssp    \
             --enable-host-pie       \
             --disable-multilib      \
             --disable-bootstrap     \
             --disable-fixincludes   \
             --with-system-zlib
```

GCC støtter syv forskjellige dataspråk, men forutsetningene for de fleste av dem er ikke installert 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 konfigureringskriptet bruker `ld` installert av `binutils` bygget tidligere i dette kapitlet, i stedet for den kryssbygde versjonen som ellers ville blitt brukt.

`--disable-fixincludes`

Som standard, under installasjonen av GCC noen systemdeklarasjoner vil være «låst» for bruk med GCC. Dette er ikke nødvendig for et moderne Linuxsystem, og potensielt skadelig hvis en pakke installeres på nytt etter installasjon av GCC. Denne bryteren hindrer GCC fra å «låse» deklarasjonene.

`--with-system-zlib`

Denne bryteren forteller GCC å koble til den systeminstallerte kopien av `zlib` biblioteket, i stedet for sin egen interne kopi.



Notat

PIE (position independent executable) er en teknikk for å produsere binære programmer som kan lastes hvor som helst i minnet. Uten PIE, sikkerhetsfunksjonen kalt ASLR (Address Space Layout Randomization) kan legges til for de delte bibliotekene, men ikke de kjørbare filene. Aktivering av PIE tillater ASLR for de kjørbare filene i tillegg til de delte bibliotekene, og reduserer noen angrep basert på faste adresser til sensitiv kode eller data i de kjørbare filene.

SSP (Stack Smashing Protection) er en teknikk for å sikre at parameterstabelen ikke er ødelagt. Stabelkorupsjon kan for eksempel endre returadressen til en subrutine, som ville tillate overføring av kontroll til en eller annen farlig kode (som eksisterer i programmet eller delte biblioteker, eller injisert av en angriper på en eller annen måte) i stedet for den originale.

Kompiler pakken:

```
make
```



Viktig

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` til **make -k check** kommandoen nedenfor hvor x er antall kjerner på systemet ditt.

GCC kan trenge mer stabelplass for å compilere noen ekstremt komplekse kodemønstre. Som en forholdsregel for vertsdistros med en liten stabelbegrensing, angi eksplisitt stabelstørrelsen hard grense til uendelig. På de fleste vertsdistroer (og det endelige LFS systemet) er den harde grensen uendelig som standard, men det er ingen skade ved å angi det eksplisitt. Det er ikke nødvendig å endre myk grense for stabelstørrelsen fordi GCC vil sette den automatisk til en passende verdi, så lenge verdien ikke overskrider den harde grensen:

```
ulimit -s -H unlimited
```

Fjern/fiks nå flere kjente testfeil:

```
sed -e '/cpython/d' -i ../gcc/testsuite/gcc.dg/plugin/plugin.exp
sed -e 's/no-pic /&-no-pie /' -i ../gcc/testsuite/gcc.target/i386/pr113689-1.c
sed -e 's/300000/(1|300000)/' -i ../libgomp/testsuite/libgomp.c-c++-common/pr109062.c
sed -e 's/{ target nonpic } //' \
-e '/GOTPCREL/d' -i ../gcc/testsuite/gcc.target/i386/fentryname3.c
```

Test resultatene som en ikke-privilegert bruker, men ikke stopp ved feil:

```
chown -R tester .
su tester -c "PATH=$PATH make -k check"
```

For å trekke ut et sammendrag av resultatene for testpakken, kjør:

```
../contrib/test_summary
```

For å filtrere ut bare sammendragene, kanaliser utdataene gjennom `grep -A7 Summ`.

Resultatene kan sammenlignes med de som ligger på <https://www.linuxfromscratch.org/lfs/build-logs/12.3/> og <https://gcc.gnu.org/ml/gcc-testresults/>.

Tsan testene er kjent for å mislykkes på enkelte vertsdistroer.

Noen få uventede feil kan ikke alltid unngås. Med mindre testresultatene er svært forskjellige fra de på URLen ovenfor, er det trygt å fortsette.

Installer pakken:

```
make install
```

GCC byggemappen 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/$(gcc -dumpmachine)/14.2.0/include{,-fixed}
```

Lag en symbolkobling som kreves av *FHS* av "historiske" grunner.

```
ln -svr /usr/bin/cpp /usr/lib
```

Mange pakker bruker navnet `cc` for å kalle C kompilatoren. Vi har allerede opprettet `cc` som en symlenke i `gcc-pass2`, opprette mansiden til den som en symbolkobling også

```
ln -sv gcc.1 /usr/share/man/man1/cc.1
```

Legg til en kompatibilitetssymbolkobling for å aktivere byggeprogrammer med optimalisering av koblingstid (LTO (Link Time Optimization)):

```
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/14.2.0/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 bør 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 -E -o '/usr/lib.*S?crt[lin].*succeeded' dummy.log
```

Utdata fra den siste kommandoen bør være:

```
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/Scrt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/crtn.o succeeded
```

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.

Bekreft at kompilatoren søker etter riktige deklarasjonsfiler:

```
grep -B4 '^ /usr/include' dummy.log
```

Denne kommandoen bør returnere følgende utdata:

```
#include <...> search starts here:
 /usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/include-fixed
 /usr/include
```

Igjen, mappen oppkalt etter måltripletten kan være annerledes enn de ovennevnte, avhengig av systemarkitekturen.

Deretter bekrefter du at den nye linkerens 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 bør utdata 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 mapper. 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");
```

Neste forsikre deg om at vi bruker riktig libc:

```
grep "/lib.*libc.so.6 " dummy.log
```

Utdata fra den siste kommandoen bør være:

```
attempt to open /usr/lib/libc.so.6 succeeded
```

Sørg for at GCC bruker riktig dynamisk linker:

```
grep found dummy.log
```

Utdataen fra den siste kommandoen bør være (som gir rom for plattformspesifikke forskjeller i dynamisk linkernavn):

```
found ld-linux-x86-64.so.2 at /usr/lib/ld-linux-x86-64.so.2
```

Hvis utdataen ikke vises som vist ovenfor eller ikke mottas i det hele tatt, så er det noe alvorlig galt. Undersøk og spor trinn for trinn for å finne ut hvor problemet er og rette det. Eventuelle 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
```

8.29.2. Innhold i GCC

Installerte programmer: `++`, `cc` (link to `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}`, `libhwasan.{a,so}`, `libitm.{a,so}`, `liblsan.{a,so}`, `liblto_plugin.so`, `libquadmath.{a,so}`, `libssp.{a,so}`, `libssp_nonshared.a`, `libstdc++.a`, `libstdc++exp.a`, `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-14.2.0`

Korte beskrivelser

c++	C++ kompilatoren
cc	C kompilatoren
cpp	C forprosessoren; den brukes av kompilatoren for å utvide <code>#include</code> , <code>#define</code> 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 for å legge til "koblingstidsoptimalisering (link time optimization)" og er ikke nyttig med standard byggealternativer
gcc-nm	En innpakning rundt nm som legger til et programtillegg til kommandolinjen. Dette programmet brukes kun for å legge til "koblingstidsoptimalisering (link time optimization)" og er ikke nyttig med standard byggealternativer
gcc-ranlib	En innpakning rundt ranlib som legger til et programtillegg til kommandolinjen. Dette programmet brukes kun for å legge til "koblingstidsoptimalisering (link time optimization)" og er ikke nyttig med standard byggealternativer
gcov	Et dekningsstestverktøy; den brukes til å analysere programmer å bestemme hvor optimaliseringer vil ha størst effekt
gcov-dump	Frakoblet (offline) <code>gcda</code> og <code>gcno</code> profildumpverktøy
gcov-tool	Frakoblet (offline) <code>gcda</code> profilbehandlingsverktøy
lto-dump	Verktøy for dumping av objektfiler produsert av GCC med LTO aktivert
<code>libasan</code>	Kjøretidsbiblioteket for adresserensing
<code>libatomic</code>	GCC atomic innebygde kjøretidsbibliotek
<code>libcc1</code>	Et bibliotek som lar GDB bruke GCC
<code>libgcc</code>	Inneholder kjøretidsstøtte for gcc
<code>libgcov</code>	Dette biblioteket er koblet inn i et program når GCC blir instruert om å aktivere profilering
<code>libgomp</code>	GNU implementering av OpenMP API for multiplattform parallellprogrammering med delt minne i C/C++ og Fortran
<code>libhwasan</code>	Maskinvareassistert kjøretidsbibliotek for Adresserensing
<code>libitm</code>	GNU transaksjonsminnebiblioteket
<code>liblsan</code>	Leak Sanitizer kjøretidsbibliotek
<code>liblto_plugin</code>	GCC sitt LTO programtillegg som lar <code>binutils</code> behandle objektfiler produsert av GCC med LTO aktivert
<code>libquadmath</code>	GCC Quad Precision Math Library API
<code>libssp</code>	Inneholder rutiner som støtter GCCs stabelknusende beskyttelsesfunksjonalitet. Normalt er det ubrukt fordi <code>glibc</code> også gir disse rutinene
<code>libstdc++</code>	Standard C++ biblioteket
<code>libstdc++exp</code>	Eksperimentelt C++ kontraktbibliotek
<code>libstdc++fs</code>	ISO/IEC TS 18822:2015 filsystembibliotek
<code>libsupc++</code>	Gir støttende rutiner for C++ programmeringsspråk
<code>libtsan</code>	Thread Sanitizer kjøretidsbibliotek

libubsan

Undefined Behavior Sanitizer kjøretidsbibliotek

8.30. Ncurses-6.5

Ncurses pakken inneholder biblioteker for terminaluavhengig håndtering av karakterskjermer.

Omtrentlig byggetid: 0.2 SBU

Nødvendig diskplass: 46 MB

8.30.1. Installasjon av Ncurses

Forbered Ncurses for kompilering:

```
./configure --prefix=/usr          \
            --mandir=/usr/share/man \
            --with-shared          \
            --without-debug        \
            --without-normal       \
            --with-cxx-shared      \
            --enable-pc-files      \
            --with-pkg-config-libdir=/usr/lib/pkgconfig
```

Betydningen av de nye konfigureringsalternativene:

--with-shared

Dette får Ncurses til å bygge og installere delte C biblioteker.

--without-normal

Dette forhindrer at Ncurses bygger og installerer statiske C biblioteker.

--without-debug

Dette forhindrer at Ncurses bygger og installerer feilsøkingsbiblioteker.

--with-cxx-shared

Dette får Ncurses til å bygge og installere delte C++ bindinger. Den forhindrer også at den bygger og installerer statiske C++ bindinger.

--enable-pc-files

Denne bryteren genererer og installerer .pc filer for pkg-config.

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.6.5`. Det kan krasje skallprosessen som bruker kode og data fra bibliotekfilen. Installer pakken med `DESTDIR`, og bytt ut bibliotekfilen riktig med **install** command (deklarasjonen `curses.h` er også redigert for å sikre ABI med wide-character skal brukes som det vi har gjort i Seksjon 6.3, «Ncurses-6.5»):

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/libncursesw.so.6.5 /usr/lib
rm -v dest/usr/lib/libncursesw.so.6.5
sed -e 's/^#if.*XOPEN.*$/#if 1/' \
     -i dest/usr/include/curses.h
cp -av dest/* /
```

Mange applikasjoner forventer fortsatt at lenkeren skal kunne finne non-wide-character Ncurses biblioteker. Lure slike applikasjoner til å koble til biblioteker med wide-character ved hjelp av symbolkoblinger (merk at `.so` lenker er bare trygt med `curses.h` redigert for alltid å bruke ABI med wide-character):

```
for lib in ncurses form panel menu ; do
  ln -sfv lib${lib}w.so /usr/lib/lib${lib}.so
  ln -sfv ${lib}w.pc /usr/lib/pkgconfig/${lib}.pc
done
```

Til slutt, sørg for at gamle programmer som ser etter `-lcurses` ved byggetiden fortsatt er byggbare:

```
ln -sfv libncursesw.so /usr/lib/libcurses.so
```

Hvis ønskelig, installer Ncurses dokumentasjonen:

```
cp -v -R doc -T /usr/share/doc/ncurses-6.5
```



Notat

Instruksjonene ovenfor oppretter ikke Ncurses med ikke-brede 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 ikke-brede 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
```

8.30.2. Innhold i Ncurses

- Installerte programmer:** `captainfo` (lenker til `tic`), `clear`, `infocmp`, `infotocap` (lenker til `tic`), `ncursesw6-config`, `reset` (lenker til `tset`), `tabs`, `tic`, `toe`, `tput`, og `tset`
- Installerte biblioteker:** `libcurses.so` (symlenke), `libform.so` (symlenke), `libformw.so`, `libmenu.so` (symlenke), `libmenuw.so`, `libncurses.so` (symlenke), `libncursesw.so`, `libncursesw.so`, `libncursesw.so`, `libpanel.so` (symlenke), and `libpanelw.so`,
- Installerte mapper:** `/usr/share/tabset`, `/usr/share/terminfo`, og `/usr/share/doc/ncurses-6.5`

Korte beskrivelser

- captainfo** 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 kildeforamt 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 rapportere det lange navnet
- tset** Kan brukes til å initialisere terminaler
- 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**

<code>libncurses++w</code>	Inneholder C++ binding for andre biblioteker i denne pakken
<code>libformw</code>	Inneholder funksjoner for å implementere skjemaer
<code>libmenuw</code>	Inneholder funksjoner for å implementere menyer
<code>libpanelw</code>	Inneholder funksjoner for å implementere paneler

8.31. Sed-4.9

Sed pakken inneholder en dataflyt (stream) redigerer.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 30 MB

8.31.1. 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 -R tester .  
su tester -c "PATH=$PATH make check"
```

Installer pakken og dens dokumentasjon:

```
make install  
install -d -m755 /usr/share/doc/sed-4.9  
install -m644 doc/sed.html /usr/share/doc/sed-4.9
```

8.31.2. Innhold i Sed

Installert program: sed
Installert mappe: /usr/share/doc/sed-4.9

Korte beskrivelser

sed Filtrerer og transformerer tekstfiler i en enkelt omgang

8.32. Psmisc-23.7

Psmisc pakken inneholder programmer for å vise informasjon om kjørende prosesser.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 6.7 MB

8.32.1. Installasjon av Psmisc

Forbered Psmisc for kompilering:

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

Kompiler pakken:

```
make
```

For å kjøre testpakken, kjør:

```
make check
```

Installer pakken:

```
make install
```

8.32.2. 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

8.33. Gettext-0.24

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: 1.7 SBU

Nødvendig diskplass: 290 MB

8.33.1. Installasjon av Gettext

Forbered Gettext for kompilering:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.24
```

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

8.33.2. 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-0.24, /usr/share/gettext, og /usr/share/gettext-0.24

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 meldingsmappe fra en oversettelsesmappe
msggrep	Trekker ut alle meldinger fra en oversettelsesmappe som samsvarer med et gitt mønster eller tilhører noen gitte kildefiler
msginit	Oppretter en ny <code>.po</code> fil, initialisere metainformasjonen med verdier fra brukerens miljø
msgmerge	Kombinerer to rå oversettelser til en 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 oversettable meldingslinjene fra den gitte kildefilen for å lage den første oversettelsesmalen
<code>libasprintf</code>	Definerer <i>autosprintf</i> klassen, som gir C formaterte utdatrutiner som kan brukes i C++ programmer, for bruk med <code><string></code> strenger og <code><iostream></code> dataflyt
<code>libgettextlib</code>	Inneholder vanlige rutiner som brukes av ulike Gettext programmer; disse er ikke beregnet for generelt bruk
<code>libgettextpo</code>	Brukes til å skrive spesialiserte programmer som behandler <code>.po</code> filer; dette biblioteket brukes når standardapplikasjonene som ble levert med Gettext (som f.eks msgcomm , msgcmp , msgattrib , og msgen) ikke er tilstrekkelig
<code>libgettextsrc</code>	Gir vanlige rutiner som brukes av ulike Gettext programmer; disse er ikke beregnet for generelt bruk
<code>libtextstyle</code>	Tekststilbibliotek
<code>preloadable_libintl</code>	Et bibliotek, ment å brukes av <code>LD_PRELOAD</code> som assisterer <code>libintl</code> i å logge uoversatte meldinger

8.34. Bison-3.8.2

Bison pakken inneholder en parsergenerator.

Omtrentlig byggetid: 2.1 SBU
Nødvendig diskplass: 62 MB

8.34.1. 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, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.34.2. 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

8.35. Grep-3.11

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

Omtrentlig byggetid: 0.4 SBU

Nødvendig diskplass: 39 MB

8.35.1. Installasjon av Grep

Fjern først en advarsel om bruk av egrep og fgrep som gjør at tester på noen pakker mislykkes:

```
sed -i "s/echo/#echo/" src/egrep.sh
```

Forbered Grep for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.35.2. Innhold i Grep

Installerte programmer: egrep, fgrep, og grep

Korte beskrivelser

- egrep** Skriver ut linjer som samsvarer med et utvidet regulært uttrykk. Den er foreldet, bruk **grep -E** i stedet
- fgrep** Skriver ut linjer som samsvarer med en liste over faste strenger Den er foreldet, bruk **grep -F** i stedet
- grep** Skriver ut linjer som samsvarer med et grunnleggende regulært uttrykk

8.36. Bash-5.2.37

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

Omtrentlig byggetid: 1.4 SBU

Nødvendig diskplass: 53 MB

8.36.1. Installasjon av Bash

Forbered Bash for kompilering:

```
./configure --prefix=/usr          \
            --without-bash-malloc  \
            --with-installed-readline \
            --docdir=/usr/share/doc/bash-5.2.37
```

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 -R tester .
```

Testpakken til pakken er designet for å kjøres som en ikke-`root` bruker som eier terminalen koblet til standardinngang. For å tilfredsstille kravet, 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
```

Testpakken bruker **diff** for å oppdage forskjellen mellom utdata fra testskriptet og forventet utdata. Enhver utdata fra **diff** (prefikset med `<` og `>`) indikerer en testfeil, med mindre det er en melding som sier at forskjellen kan ignoreres. En test med navnet `run-builtins` er kjent for å mislykkes på noen vertsdistroer med en forskjell på den første linjen i utdataen.

Installer pakken:

```
make install
```

Kjør den nylig kompilerte **bash** programmet (erstatter det som kjøres for øyeblikket):

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

8.36.2. Innholdet i Bash

Installerte programmer: `bash`, `bashbug`, og `sh` (link to `bash`)

Installerte mapper: `/usr/include/bash`, `/usr/lib/bash`, og `/usr/share/doc/bash-5.2.37`

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 skalkript 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**, prøver **bash** å etterligne oppstartadferd av historiske versjoner av **sh** så nært som mulig, samtidig som den også samsvarer med POSIX standarden

8.37. Libtool-2.5.4

Libtool pakken inneholder GNU generiske bibliotekstøtteskript. Det omslutter kompleksiteten ved å bruke delte biblioteker i en konsistent, overførbart grensesnitt.

Omtrentlig byggetid: 0.6 SBU

Nødvendig diskplass: 44 MB

8.37.1. Installasjon av Libtool

Forbered Libtool 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/libltdl.a
```

8.37.2. 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 ulike vanskelighetene med å åpne dynamisk lastede biblioteker

8.38. GDBM-1.24

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: less than 0.1 SBU
Nødvendig diskplass: 13 MB

8.38.1. 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 eldre DBM rutiner det gir.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.38.2. 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 modifierer en GDBM database
libgdbm	Inneholder funksjoner for å manipulere en hashet database
libgdbm_compat	Kompatibilitetsbibliotek som inneholder eldre DBM funksjoner

8.39. Gperf-3.1

Gperf genererer en perfekt hashfunksjon fra et nøkkelsett.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 6.1 MB

8.39.1. 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
```

8.39.2. Innhold i Gperf

Installert program: gperf
Installert mappe: /usr/share/doc/gperf-3.1

Korte beskrivelser

gperf Genererer en perfekt hash fra et nøkkelsett

8.40. Expat-2.6.4

Expat pakken inneholder et dataflytorientert C bibliotek for å analysere XML.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 14 MB

8.40.1. Installasjon av Expat

Forbered Expat for kompilering:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.6.4
```

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-2.6.4
```

8.40.2. Innhold i Expat

Installert program: xmlwf
Installerte biblioteker: libexpat.so
Installert mappe: /usr/share/doc/expat-2.6.4

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

8.41. Inetutils-2.6

Inetutils pakken inneholder programmer for grunnleggende nettverksbygging.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 35 MB

8.41.1. Installasjon av Inetutils

Først, gjør sånn at pakken bygget med gcc-14.1 eller senere:

```
sed -i 's/def HAVE_TERMCAP_TGETENT/ 1/' telnet/telnet.c
```

Forbered Inetutils for kompilering:

```
./configure --prefix=/usr \
            --bindir=/usr/bin \
            --localstatedir=/var \
            --disable-logger \
            --disable-whois \
            --disable-rpc \
            --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 openssl 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
```

8.41.2. 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 nettverksgrensesnitt
ping	Sender ekkoforespørselspakker og rapporterer hvor lang tid 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

8.42. Less-668

Less pakken inneholder et tekstfilviser.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 14 MB

8.42.1. 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
```

For å teste resultatene, kjør:

```
make check
```

Installer pakken:

```
make install
```

8.42.2. 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 tastaturlbindingene for less

8.43. Perl-5.40.1

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

Omtrentlig byggetid: 1.3 SBU

Nødvendig diskplass: 245 MB

8.43.1. Installasjon av Perl

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 \
-D prefix=/usr \
-D vendorprefix=/usr \
-D privlib=/usr/lib/perl5/5.40/core_perl \
-D archlib=/usr/lib/perl5/5.40/core_perl \
-D sitelib=/usr/lib/perl5/5.40/site_perl \
-D sitearch=/usr/lib/perl5/5.40/site_perl \
-D vendorlib=/usr/lib/perl5/5.40/vendor_perl \
-D vendorarch=/usr/lib/perl5/5.40/vendor_perl \
-D man1dir=/usr/share/man/man1 \
-D man3dir=/usr/share/man/man3 \
-D pager="/usr/bin/less -isR" \
-D useshrplib \
-D usethreads
```

Betydningen av de nye konfigureringsalternativene:

```
-D pager="/usr/bin/less -isR"
```

Dette sikrer at `less` brukes i stedet for `more`.

```
-D man1dir=/usr/share/man/man1 -D man3dir=/usr/share/man/man3
```

Siden Groff ikke er installert ennå vil ikke **Configure** opprette mansider for Perl. Disse parameterene overstyrrer denne oppførselen.

```
-D usethreads
```

Bygg perl med støtte for tråder.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
TEST_JOBS=$(nproc) make test_harness
```

Installer pakken og rydd opp:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

8.43.2. Innhold i Perl

Installerte programmer:	corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.40.1 (hard lenke til perl), perlbug, perldoc, perlivp, perlthanks (hard lenke til 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 modulen
perl	Kombinerer noen av de beste egenskapene til C, sed , awk og sh til et singelt swiss-army språk
perl5.40.1	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 sjekksommer
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

8.44. XML::Parser-2.47

XML::Parser modulen er et Perl grensesnitt til James Clarks XML-parser, Expat.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 2.4 MB

8.44.1. 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
```

8.44.2. Innhold i XML::Parser

Installert modul: Expat.so

Korte beskrivelser

Expat gir Perl Expat grensesnittet

8.45. Intltool-0.51.0

Intltool er et internasjonalsiseringsverktøy som brukes til å trekke ut oversettbare strenger fra kildefiler.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 1.5 MB

8.45.1. Installasjon av Intltool

Rett først en advarsel som er forårsaket av perl-5.22 og senere:

```
sed -i 's:\\\\${:\\\\$\\{: ' intltool-update.in
```



Notat

Det regulære uttrykket ovenfor ser uvanlig ut på grunn av alle skråstreker. Det den gjør er å legge til en 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
```

8.45.2. 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

8.46. Autoconf-2.72

Autoconf pakken inneholder programmer for å produsere skallskript som automatisk kan konfigurere kildekoden.

Omtrentlig byggetid: less than 0.1 SBU (about 0.4 SBU with tests)
Nødvendig diskplass: 25 MB

8.46.1. Installasjon av Autoconf

Forbered Autoconf for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.46.2. 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 <i>#define</i> uttrykk for configure å bruke
autom4te	En innpakning for M4 makroprosessen
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 <code>configure.in</code> fil for en programvarepakke; den undersøker kildefilene i et mappetre, slker etter vanlige problemer med portabilitet, og oppretter en <code>configure.scan</code> fil som fungerer som en innledende <code>configure.in</code> fil for en pakke
autoupdate	Endrer en <code>configure.in</code> fil som fortsatt anroper autoconf makroer ved deres gamle navn til å bruke gjeldende makronavn
ifnames	Hjelper når det skrives <code>configure.in</code> filer for en programvarepakke; den skriver ut identifikatorene som pakken bruker i C forbehandlerbetingelser [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 <code>configure.in</code> fil generert av autoscan .]

8.47. Automake-1.17

Automake pakken inneholder programmer for å generere Make filer for bruk med Autoconf.

Omtrentlig byggetid: less than 0.1 SBU (about 1.1 SBU with tests)
Nødvendig diskplass: 121 MB

8.47.1. Installasjon av Automake

Forbered Automake for kompilering:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.17
```

Kompiler pakken:

```
make
```

Å bruke fire parallelle jobber øker hastigheten på testene, selv på systemer med mindre logiske kjerner, på grunn av interne forsinkelser i individuelle tester. Å teste resultatene, utsted:

```
make -j$(($(nproc)>4?$(nproc):4)) check
```

Erstatt $\$(\dots)$ med antall logiske kjerner du vil bruke, hvis du ikke vil bruke alle.

Installer pakken:

```
make install
```

8.47.2. Innholdet i Automake

Installerte programmer: aclocal, aclocal-1.17 (hardlinket til aclocal), automake, og automake-1.17 (hardlinket til automake)

Installerte mapper: /usr/share/aclocal-1.17, /usr/share/automake-1.17, og /usr/share/doc/automake-1.17

Korte beskrivelser

aclocal	Genererer <code>aclocal.m4</code> filer basert på innholdet i <code>configure.in</code> filene
aclocal-1.17	En hard lenke til aclocal
automake	Et verktøy for automatisk generering av <code>Makefile.in</code> filer fra <code>Makefile.am</code> filer [For å lage alle <code>Makefile.in</code> filer for en pakke, kjør dette programmet i mappen på øverste nivå. Ved å skanne <code>configure.in</code> filen, finner den automatisk hver passende <code>Makefile.am</code> fil og genererer tilsvarende <code>Makefile.in</code> fil.]
automake-1.17	En hard lenke til automake

8.48. OpenSSL-3.4.1

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: 1.8 SBU
Nødvendig diskplass: 920 MB

8.48.1. 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:

```
HARNESS_JOBS=$(nproc) make test
```

En test, 30-test_afalg.t, er kjent for å mislykkes hvis vertskjernen ikke har `CONFIG_CRYPT_USER_API_SKCIPHER` aktivert, eller ikke har noen alternativer som gir en AES med CBC implementering (for eksempel en kombinasjonen av `CONFIG_CRYPT_AES` og `CONFIG_CRYPT_CBC`, eller `CONFIG_CRYPT_AES_NI_INTEL` hvis CPU støtter AES-NI) aktivert. Hvis den mislykkes, kan den trygt ignoreres.

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-3.4.1
```

Hvis ønskelig, installer litt tilleggsdokumentasjon:

```
cp -vfr doc/* /usr/share/doc/openssl-3.4.1
```



Notat

Du bør oppdatere OpenSSL når en ny versjon som fikser sårbarheter er annonsert. Siden OpenSSL 3.0.0, OpenSSL-versjonsordningen følger MAJOR.MINOR.PATCH-formatet. API/ABI-kompatibilitet er garantert for samme MAJOR versjonsnummer. Fordi LFS installerer kun de delte bibliotekene, er det ikke nødvendig å recompile pakker som lenker til `libcrypto.so` eller `libssl.so` ved oppgradering til en versjon med uendret MAJOR versjonsnummer.

Imidlertid må alle kjørende programmer koblet til disse bibliotekene stoppes og startes på nytt. Les de relaterte oppføringene i Seksjon 8.2.1, «Oppgraderingsproblemer» for detaljer.

8.48.2. 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-3.4.1`

Korte beskrivelser

c_rehash	er et Perl skript som skanner alle filer i en katalog og legger til symbolske lenker til deres hashverdier. Bruk av c_rehash vurderes foreldet og bør erstattes av openssl rehash kommandoen
openssl	er et kommandolinjeverktøy for bruk av de ulike kryptografifunksjonene til OpenSSL sitt kryptobibliotek fra skallet. Den kan brukes til ulike funksjoner som er dokumentert i <i>openssl(1)</i>
libcrypto.so	implementerer et bredt spekter av kryptografiske algoritmer som brukes i ulike Internettstandarder. 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, dokumentasjonen kan bli funnet i <i>ssl(7)</i>

8.49. Libelf fra Elfutils-0.192

Libelf er et bibliotek for håndtering av ELF (kjørbare og linkbare formater) filer.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 135 MB

8.49.1. Installasjon av Libelf

Libelf er en del av elfutils-0.192 pakken. Bruk elfutils-0.192.tar.bz2 filen som kildetarball.

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
```

8.49.2. Innhold i Libelf

Installert bibliotek: libelf.so
Installert mappe: /usr/include/elfutils

Korte beskrivelser

libelf.so Inneholder API funksjoner for å håndtere ELF objektfiler

8.50. Libffi-3.4.7

Libffi-biblioteket gir en portabel høynivå programmeringsgrensesnitt til ulike kallkonvensjoner. Dette lar en programmerer kalle enhver funksjon ved kjøretid, spesifisert av et kallgrensesnittbeskrivelse.

FFI står for Foreign Function Interface. En FFI tillater et program skrevet på ett språk å kalle et program skrevet på et annet språk. Nærmere bestemt, Libffi kan gi en bro mellom en tolk som Perl, eller Python, og delte biblioteksunderrutiner skrevet i C eller C++.

Omtrentlig byggetid: 1.7 SBU
Nødvendig diskplass: 11 MB

8.50.1. Installasjon av Libffi



Notat

I likhet med GMP bygges libffi med spesifikke optimaliseringer til prosessoren som er i bruk. Hvis du bygger for et annet system, endre verdien av `--with-gcc-arch=` parameteren i følgende kommando til et arkitekturnavn fullt implementert av CPU på det systemet. Hvis dette ikke gjøres, vil alle applikasjoner som lenker til `libffi` utløse ulovlige operasjonsfeil (Illegal Operation Errors).

Forbered libffi for kompilering:

```
./configure --prefix=/usr \
            --disable-static \
            --with-gcc-arch=native
```

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 er kanskje ikke riktig. Hvis den genererte koden vil bli kopiert fra det opprinnelige systemet til et mindre kapabelt system, bruk det mindre kapable systemet som parameter. For detaljer om alternative systemtyper, se *x86 alternativene i GCC manualen*.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.50.2. Innhold i Libffi

Installert bibliotek: libffi.so

Korte beskrivelser

`libffi` Inneholder API funksjonene for fremmede funksjonsgrensesnitt

8.51. Python-3.13.2

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

Omtrentlig byggetid: 2.1 SBU

Nødvendig diskplass: 501 MB

8.51.1. Installasjon av Python 3

Forbered Python for kompilering:

```
./configure --prefix=/usr \
            --enable-shared \
            --with-system-expat \
            --enable-optimizations
```

Betydningen av konfigureringsalternativene:

--with-system-expat

Denne bryteren muliggjør kobling mot systemversjonen av Expat.

--enable-optimizations

Denne bryteren muliggjør omfattende, men tidkrevende, optimaliseringstrinn. Tolken bygges to ganger; tester utført på det første bygget brukes til å forbedre den optimaliserte endelige versjonen.

Kompiler pakken:

```
make
```

Noen tester er kjent for iblant å henge på ubestemt tid. Så for å teste resultatet, kjør testpakken, men sett en tidsbegrensning på 2 minutter for hvert testforsøk:

```
make test TESTOPTS="--timeout 120"
```

For et relativt tregt system må du kanskje øke tidsgrensen og 1 SBU (målt når du bygger Binutils passerer 1 med én CPU kjerne) bør være nok. Noen tester er rare, så testpakken vil automatisk kjøre mislykkede tester på nytt. Hvis en test mislyktes, men deretter består når den kjøres på nytt, bør den anses som bestått. En test, `test_ssl`, er kjent for å mislykkes i chroot miljøet.

Installer pakken:

```
make install
```

Vi bruker **pip3** kommandoen til å installere Python 3 programmer og moduler for alle brukere som `root` flere steder i denne boken. Dette er i konflikt med Python utviklernes anbefaling: å installere pakker i et virtuelt miljø eller inn i hjemmemappen til en vanlig bruker (ved å kjøre **pip3** som denne brukeren). En advarsel med flere linjer utløses når **pip3** er utstedt av `root` brukeren.

Hovedgrunnen for anbefalingen er å unngå konflikter med systemets pakkeansvarlig (**dpkg**, for eksempel). LFS har ikke en systemomfattende pakkebehandling, så dette er ikke et problem. Også **pip3** vil se etter en ny versjon av seg selv når den kjøres. Siden domenenavnoppløsning ikke er konfigurert ennå i LFS chroot miljøet, **pip3** kan ikke sjekke for en ny versjon av seg selv, og vil produsere en advarsel.

Etter at vi har startet opp LFS systemet og satt opp en nettverkstilkobling, en annen advarsel vil bli gitt, og ber brukeren om å oppdatere **pip3** fra et forhåndsbygd wheel på PyPI (når en ny versjon er tilgjengelig). Men LFS vurderer **pip3** å være en del av Python 3, så det burde ikke oppdateres separat. Dessuten vil en oppdatering fra et

forhåndsbygd wheel avvike fra vårt mål: å bygge et Linuxsystem fra kildekoden. Så advarsel om en ny versjon av **pip3** bør ignoreres om vi vil. Hvis du ønsker det, kan du undertrykke alle disse advarslene ved å kjøre følgende kommando, som oppretter en konfigurasjonsfil:

```
cat > /etc/pip.conf << EOF
[global]
root-user-action = ignore
disable-pip-version-check = true
EOF
```



Viktig

I LFS og BLFS bygger og installerer vi normalt Pythonmoduler med **pip3** kommandoen. Vennligst pass på at **pip3 install** kommandoer i begge bøkene kjøres som `root` brukeren med mindre det er for et virtuelt Python-miljø å kjøre en **pip3 install** som en ikke-`root` bruker kan synes å fungerer fint, men det vil føre til at den installerte modulen blir utilgjengelig av andre brukere.

pip3 install vil ikke installere en allerede installert modul som standard. Når du bruker **pip3 install** kommandoen for å oppgradere en modul (for eksempel fra meson-0.61.3 til meson-0.62.0), sett inn alternativet `--upgrade` inn i kommandolinjen. Hvis det virkelig er nødvendig å nedgradere en modul, eller installer samme versjon på nytt av en eller annen grunn, sett inn `--force-reinstall --no-deps` inn i kommandolinjen.

Hvis ønskelig, installer den forhåndsformaterte dokumentasjonen:

```
install -v -dm755 /usr/share/doc/python-3.13.2/html

tar --strip-components=1 \
  --no-same-owner \
  --no-same-permissions \
  -C /usr/share/doc/python-3.13.2/html \
  -xvf ../python-3.13.2-docs-html.tar.bz2
```

Betydningen av dokumentasjonsinstallasjons kommandoene:

`--no-same-owner` Og `--no-same-permissions`

Sørger for at de installerte filene har riktig eierskap og tillatelser. Uten disse alternativene, tar vil installere pakkefilene med oppstrømsskaperens verdier.

8.51.2. Innhold i Python 3

Installerte programmer: 2to3, idle3, pip3, pydoc3, python3, og python3-config

Installert bibliotek: libpython3.13.so og libpython3.so

Installerte mapper: /usr/include/python3.13, /usr/lib/python3, og /usr/share/doc/python-3.13.2

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 et Python bevisst GUI tekstprogram. For at dette skriptet skal kjøre, må du ha installert Tk før Python slik at Tkinter Pythonmodulen 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 tolken for Python, et tolket, interaktiv, objektorientert programmeringsspråk

8.52. Flit-Core-3.11.0

Flit-core er den distribusjonsbyggende delene av Flit (et pakkeverktøy for enkle Pythonmoduler).

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 1.0 MB

8.52.1. Installasjon av Flit-Core

Bygg pakken:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Installerer pakken:

```
pip3 install --no-index --find-links dist flit_core
```

Betydningen av pip3 konfigurasjonsalternativene og kommandoene:

wheel

Denne kommandoen bygger wheelarkivet for denne pakken.

-w dist

Instruerer pip å putte det opprettede wheellet i `dist` mappen.

--no-cache-dir

Hindrer pip fra å kopiere det opprettede wheel inn i `/root/.cache/pip` mappen.

install

Denne kommandoen installerer pakken.

--no-build-isolation, --no-deps, and --no-index

Disse alternativene forhindrer henting av filer fra nettets pakkerepository (PyPI). Hvis pakkene er installert i riktig rekkefølge, trenger ikke pip å hente noen filer i utgangspunktet; disse alternativer gir en viss sikkerhet i tilfelle brukerfeil.

--find-links dist

Instruerer pip til å søke etter wheelarkiver i `dist` mappen.

8.52.2. Innhold i Flit-Core

Installert mappe: `/usr/lib/python3.13/site-packages/flit_core` og `/usr/lib/python3.13/site-packages/flit_core-3.11.0.dist-info`

8.53. Wheel-0.45.1

Wheel er et Python bibliotek som er referanseimplementeringen av Python wheel pakkestandarden.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 1.6 MB

8.53.1. Installasjon av Wheel

Kompiler Wheel med følgende kommando:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Installer wheel med følgende kommando:

```
pip3 install --no-index --find-links dist wheel
```

8.53.2. Innholdet i Wheel

Installert program: wheel
Installerte mapper: /usr/lib/python3.13/site-packages/wheel og /usr/lib/python3.13/site-packages/wheel-0.45.1.dist-info

Kort beskrivelse

wheel er et verktøy for å pakke ut, pakke eller konvertere wheelarkiver

8.54. Setuptools-75.8.1

Setuptools er et verktøy som brukes til å laste ned, bygge, installere, oppgradere, og avinstallerer Python pakker.

Omtrentlig byggetid: less than 0.1 SBU

Nødvendig diskplass: 26 MB

8.54.1. Installasjon av Setuptools

Bygg pakken:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Installer pakken:

```
pip3 install --no-index --find-links dist setuptools
```

8.54.2. Innhold i Setuptools

Installert mappe: /usr/lib/python3.13/site-packages/_distutils_hack, /usr/lib/python3.13/site-packages/pkg_resources, /usr/lib/python3.13/site-packages/setuptools, and /usr/lib/python3.13/site-packages/setuptools-75.8.1.dist-info

8.55. Ninja-1.12.1

Ninja er et lite byggesystem med fokus på hastighet.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 37 MB

8.55.1. 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 bruke opp systemets minne. Når **ninja** påkalles fra kommandolinjen, å sende parameteren `-jN` vil begrense antall parallelle prosesser. Noen pakker legger inn utførelsen av **ninja**, og gir ikke parameteren `-j` videre til den.

Ved å bruke den *valgfrie* prosedyren nedenfor lar en bruker begrense antall parallelle prosesser via en miljøvariabel, **NINJAJOBS**. **For eksempe**, å sette:

```
export NINJAJOBS=4
```

vil begrense **ninja** til fire parallelle prosesser.

Om ønskelig, la **ninja** gjenkjenne miljøvariabelen **NINJAJOBS** ved å kjøre strømredigeringsprogrammet:

```
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 --verbose
```

Betydningen av byggealternativet:

--bootstrap

Denne parameteren tvinger **ninja** til å gjenoppbygge seg selv for gjeldene system.

--verbose

Denne parameteren gjør at **configure.py** viser fremdriften under bygging av Ninja.

Pakketestene kan ikke kjøres i chroot miljøet. Det krever *cmake*. Men det grunnleggende funksjonen til denne pakken er allerede testet ved å gjenoppbygge seg selv (med *--bootstrap* alternativet).

Install the package:

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
```

8.55.2. Innhold av Ninja

Installerte programmer: `ninja`

Korte beskrivelser

ninja er Ninja byggesystemet

8.56. Meson-1.7.0

Meson er et åpen kildekode byggesystem ment å være både ekstremt raskt og så brukervennlig som mulig.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 44 MB

8.56.1. Installasjon av Meson

Kompiler Meson med følgende kommando:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Testpakken krever noen pakker utenfor omfanget av LFS.

Installer pakken:

```
pip3 install --no-index --find-links dist meson
install -vDm644 data/shell-completions/bash/meson /usr/share/bash-completion/completions/meson
install -vDm644 data/shell-completions/zsh/_meson /usr/share/zsh/site-functions/_meson
```

Betydningen av installasjonsparametrene:

-w dist

Putter det opprettede wheels inn i `dist` mappen.

--find-links dist

Installerer wheels fra `dist` mappen.

8.56.2. Innhold i Meson

Installerte programmer: meson
Installert mappe: /usr/lib/python3.13/site-packages/meson-1.7.0.dist-info og /usr/lib/python3.13/site-packages/mesonbuild

Korte beskrivelser

meson Et byggesystem med høy produktivitet

8.57. Kmod-34

Kmod pakken inneholder biblioteker og verktøy for lastning av kjernemoduler

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 11 MB

8.57.1. Installasjon av Kmod

Forbered Kmod for kompilering:

```
mkdir -p build
cd      build

meson setup --prefix=/usr .. \
           --sbindir=/usr/sbin \
           --buildtype=release \
           -D manpages=false
```

Betydningen av konfigureringsalternativene:

-D manpages=false

Dette alternativet deaktiverer generering av manualsider som krever et eksternt program.

Kompiler pakken:

```
ninja
```

Testpakken til denne pakken krever rå kjernedeklarasjoner (ikke de «sanitiserte» kjernedeklarasjonene installert tidligere), som er utenfor rammen av LFS.

Installer nå pakken:

```
ninja install
```

8.57.2. 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 modulene

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

8.58. Coreutils-9.6

Coreutils pakken inneholder de grunnleggende hjelpeprogrammene som trengs av hvert operativsystem.

Omtrentlig byggetid: 1.2 SBU

Nødvendig diskplass: 182 MB

8.58.1. Installasjon av Coreutils

POSIX krever at programmer fra Coreutils gjenkjenner karaktergrenser riktig selv i multibyte lokaliteter. Følgende oppdateringer fikser dette misligholdet og andre internasjonaliseringsrelaterte feil.

```
patch -Np1 -i ../coreutils-9.6-i18n-1.patch
```



Notat

Tidligere ble det funnet mange feil i denne oppdateringen. Ved melding om nye feil til Coreutils vedlikeholdere, vennligst først sjekk om de er reproduerbare uten denne oppdateringen.

Forbered nå Coreutils for kompilering:

```
autoreconf -fv
automake -af
FORCE_UNSAFE_CONFIGURE=1 ./configure \
    --prefix=/usr \
    --enable-no-install-program=kill,uptime
```

Betydningen av kommandoene og konfigureringsalternativene:

autoreconf -fv

Oppdateringen for internasjonalisering har modifisert byggesystemet til pakken, slik at konfigurasjonsfilene må bli regenerert. Normalt ville vi brukt `-i` alternativet for å oppdatere standard hjelpefiler, men for denne pakken fungerer det ikke fordi `configure.ac` spesifiserte en gammel gettext versjon.

automake -af

Automake hjelpefilerne ble ikke oppdatert av **autoreconf** på grunn av savnet `-i` alternativ. Denne kommandoen oppdaterer dem for å forhindre byggefeil.

`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 å installere programmer 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 en gruppe. Sånn at disse testene ikke hoppes over, legg til en midlertidig gruppe og gjør bruker `tester` en del av de:

```
groupadd -g 102 dummy -U tester
```

Fiks noen av tillatelsene slik at ikke-`root` brukeren kan compilere og kjøre testene:

```
chown -R tester .
```

Kjør nå testene (ved hjelp av `/dev/null` for standard inngang, ellers kan to tester bli brutt hvis du bygger LFS i en grafisk terminal eller en økt i SSH eller GNU Skjerm fordi standard inngang er koblet til en PTY fra vertsdistro, og enhetsnoden for en slik PTY kan ikke nås fra LFS chroot miljøet):

```
su tester -c "PATH=$PATH make -k RUN_EXPENSIVE_TESTS=yes check" \  
< /dev/null
```

Fjern den midlertidige gruppen:

```
groupdel dummy
```

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
```

8.58.2. 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) sjekksommer

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 utgang

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 mapper

chroot Kjører en kommando med den angitte mappen som / mappen

cksum Skriver ut sjekksommen 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 dato og klokkeslett i det gitte formatet, eller stiller inn systemdato og klokkeslett
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 mappe (det samme som ls kommandoen)
dircolors	Skriver ut kommandoer for å angi <code>LS_COLOR</code> miljøvariabelen for å endre fargeskjemaet som brukes av ls
dirname	Trekker ut mappdelen(e) av gitte navn
du	Rapporterer hvor mye diskplass som brukes av gjeldende mappe, av hver av de gitte mappene (inkludert alle undermapper) 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 de spesifiserte heltallene
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 lenke (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 mappe
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 mapper
nice	Kjører et program med endret planleggingsprioritet
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 oktal og andre formater
paste	Slår sammen de gitte filene og kobler sammen sekvensielt tilsvarende linjer side ved side, atskilt med tabulator tegn
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 arbeidsmappe
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 kontrollerer 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 blokketellinger 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 fildidsstempler, angir tilgang og endringstider 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 den delvise rekkefølgen 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 en 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 det totale linjer når mer enn en fil er gitt
who	Rapporterer hvem som er pålogget
whoami	Rapporterer brukernavnet som er knyttet til gjeldende effektive bruker-ID
yes	Skriver ut <i>y</i> , gjentatte ganger eller en gitt streng til den drepes
<code>libstdbuf</code>	Bibliotek brukt av stdbuf

8.59. Check-0.15.2

Check er et rammeverk for C enhetstesting.

Omtrentlig byggetid: 0.1 SBU (about 2.1 SBU with tests)
Nødvendig diskplass: 11 MB

8.59.1. 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
```

8.59.2. Innholdet i Check

Installert program: checkmk
Installert bibliotek: libcheck.so

Korte beskrivelser

checkmk Awk skript for å generere C enhetstester for bruk med Check sitt rammeverk for enhetstesting
libcheck.so Inneholder funksjoner som gjør at Check kan kalles fra et test program

8.60. Diffutils-3.11

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

Omtrentlig byggetid: 0.4 SBU

Nødvendig diskplass: 50 MB

8.60.1. Installasjon av Diffutils

Forbered Diffutils for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.60.2. Innhold i Diffutils

Installerte programmer: cmp, diff, diff3, og sdiff

Korte beskrivelser

- cmp** Sammenligner to filer og rapporterer eventuelle forskjeller byte for byte
- 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

8.61. Gawk-5.3.1

Gawk pakken inneholder programmer for å manipulere tekstfiler.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 43 MB

8.61.1. 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:

```
chown -R tester .  
su tester -c "PATH=$PATH make check"
```

Installer pakken:

```
rm -f /usr/bin/gawk-5.3.1  
make install
```

Betydningen av parameter:

rm -f /usr/bin/gawk-5.3.1

Byggesystemet vil ikke gjenskape den harde lenken `gawk-5.3.1` hvis den allerede eksisterer. Fjern den for å sikre at den forrige hardlinken installert i Seksjon 6.9, «Gawk-5.3.1» er oppdatert her.

Installasjonsprosessen har allerede opprettet **awk** som en symlenke til **gawk**, opprette mandsiden til den som en symlenke også

```
ln -sv gawk.1 /usr/share/man/man1/awk.1
```

Hvis ønskelig, installer dokumentasjonen:

```
install -vDm644 doc/{awkforai.txt,*.eps,pdf,jpg} -t /usr/share/doc/gawk-5.3.1
```

8.61.2. Innhold i Gawk

Installerte programmer: awk (lenker til gawk), gawk, og gawk-5.3.1

Installerte biblioteker: filefuncs.so, fnmatch.so, fork.so, inplace.so, intdiv.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwo-way.so, rvarray.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-5.3.1

Korte beskrivelser

awk En lenke til **gawk**

gawk Et program for å manipulere tekstfiler; det er GNU implementeringen av **awk**

gawk-5.3.1 En hard lenke til **gawk**

8.62. Findutils-4.10.0

Findutils pakken inneholder programmer for å finne filer. Programmer er tilgjengelig for å søke gjennom alle filene i et katalogtre og til opprette, vedlikeholde og søke i en database (ofte raskere enn den rekursive find, men upålitelig med mindre databasen nylig har blitt oppdatert). Findutils leverer også **xargs** programmet, som kan brukes til å kjøre en spesifisert kommando på hver fil valgt av et søk.

Omtrentlig byggetid: 0.7 SBU

Nødvendig diskplass: 63 MB

8.62.1. Installasjon av Findutils

Forbered Findutils for kompilering:

```
./configure --prefix=/usr --localstatedir=/var/lib/locate
```

Betydningen av konfigureringsalternativene:

```
--localstatedir
```

Dette alternativet flytter **locate** databasen til `/var/lib/locate`, som er den FHS kompatible plasseringen.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Installer pakken:

```
make install
```

8.62.2. Innhold i Findutils

Installerte programmer: find, locate, updatedb, og xargs

Installert mappe: /var/lib/locate

Korte beskrivelser

find	Søker i gitte mappetrær 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

8.63. Groff-1.23.0

Groff pakken inneholder programmer for prosessering og formatering av tekst.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 108 MB

8.63.1. Installasjon av Groff

Groff forventer at miljøvariabelen `PAGE` inneholder 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
```

Bygg pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.63.2. 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-1.23.0, /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 forbehandler for å lage kjemiske strukturdiagrammer
eqn	Kompilerer beskrivelser av ligninger innebygd i troff inndatafiler 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	Forbehandler for groff, tillater tillegg av perl kode inn i groff filer
gpinyin	Forbehandler for groff, tillater innsetting av Pinyin (Mandarin-kinesisk stavet med det romerske alfabetet) til groff filer.
grap2graph	Konverterer et grapdiagram til et beskåret punktgrafikkbilde (grap er et gammelt Unix-programmeringsspråk for å lage diagrammer)

grn	En groff forbehandler for gremlin filer
grodvi	En driver for groff som produserer utdatafiler med TeX dvi formatet
groff	En grenseflate til groff dokumentformateringsystemet; normalt, kjøres troff programmet og en etterbehandler passende for den valgte enheten
groffer	Viser groff filer og mansider på X og tty terminaler
grog	Leser filer og gjetter hvilket av groff alternativene <code>-e</code> , <code>-man</code> , <code>-me</code> , <code>-mm</code> , <code>-ms</code> , <code>-p</code> , <code>-s</code> , og <code>-t</code> kreves for å skrive ut filer, og rapporterer groff kommandoen inkludert de 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 fra GNU troff til PDF
grops	Oversetter utdataene fra GNU troff til PostScript
grotty	Oversetter utdataene fra 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 forbehandler for groff
neqn	Formaterer ligninger for amerikansk standardkode for informasjon utveksling (ASCII) utdata
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 <code>.pfb</code> 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 fra GNU troff til HTML
preconv	Konverterer koding av inndatafiler til noe GNU troff forstår
pre-grohtml	Oversetter utdataene fra GNU troff til HTML
refer	Kopierer innholdet i en fil til standardutgang, unntatt linjene mellom <code>./</code> og <code>./</code> som tolkes som referanser, og linjer mellom <code>.R1</code> og <code>.R2</code> som 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 PDF filer

roff2ps	Transformerer Roff filer til ps filer
roff2text	Transformerer Roff filer til tekst filer
roff2x	Transformerer Roff filer til andre formater
soelim	Leser filer og erstatter linjer i formatet <i>.so file</i> av innholdet i nevnte <i>file</i>
tbl	Kompilerer beskrivelser av tabeller innebygd i troff inndatafiler til kommandoer som forstås av troff
tfmtoedit	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 forbehandler og etterbehandler i riktig rekkefølge og med passende alternativer

8.64. GRUB-2.12

GRUB pakken inneholder en oppstartslaster (GRand Unified Bootloader).

Omtrentlig byggetid: 0.3 SBU

Nødvendig diskplass: 166 MB

8.64.1. Installasjon av GRUB



Notat

Hvis systemet ditt har UEFI støtte og du ønsker å starte LFS med UEFI, du må installere GRUB med UEFI støtte (og dets avhengigheter) med å følge instruksjonene på *BLFS siden*. Du kan hoppe over denne pakken, eller installere denne pakken og BLFS GRUB for UEFI pakken uten konflikt (BLFS siden gir instruksjoner for begge tilfeller).



Advarsel

Fjern eventuelle miljøvariabler som kan påvirke bygget:

```
unset {C,CPP,CXX,LD}FLAGS
```

Ikke prøv å «optimalisere» denne pakken med tilpassete kompileringsflagg. Denne pakken er en oppstartslaster. Da kan lavnivå operasjonene i kildekoden bli brutt av aggressiv optimalisering.

Legg til en fil som mangler fra utgivelsens tarball:

```
echo depends bli part_gpt > grub-core/extra_deps.lst
```

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 eliminere noen 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 begrensede LFS miljøet. For å kjøre testene uansett, kjør **make check**.

Installer pakken, og flytt støttefilen for Bash fullføring til plasseringen anbefalt av Bash fullføringsvedlikeholderne:

```
make install
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

Å gjøre LFS systemet oppstartbart med GRUB vil bli diskutert i Seksjon 10.4, «Bruke GRUB til å sette opp oppstartprosessen»

8.64.2. Innhold i GRUB

Installerte programmer: grub-bios-setup, grub-editenv, grub-file, grub-fstest, grub-glue-efi, grub-install, grub-kbdcomp, grub-macbless, grub-menu.lst2cfg, grub-mkconfig, grub-mkimage, grub-mklayout, grub-mknetdir, grub-mkpasswd-pbkdf2, grub-mkrelpath, grub-mkrescue, grub-mkstandalone, grub-ofpathname, grub-probe, grub-reboot, grub-render-label, grub-script-check, grub-set-default, grub-sparc64-setup, 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 den gitte filen er av den angitte typen
grub-fstest	Verktøy for å feilsøke fil system driveren
grub-glue-efi	Limer 32-biters og 64-biters binære filer til en enkelt fil (for Apple maskiner)
grub-install	Installer GRUB på harddisken din
grub-kbdcomp	Skript som konverterer et xkb oppsett til et som gjenkjennes av GRUB
grub-macbless	Er Mac-style bless for HFS eller HFS+ filsystemer (bless er særegen for Apple-maskiner; det gjør en enhet oppstartbar)
grub-menu.lst2cfg	Konverterer en GRUB Legacy <code>menu.lst</code> til en <code>grub.cfg</code> for bruk med GRUB 2
grub-mkconfig	Generer en <code>grub.cfg</code> fil
grub-mkimage	Lager et oppstartbart bilde av GRUB
grub-mklayout	Genererer en GRUB tastaturopsettfil
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, CDROM/DVD, eller en USB stasjon
grub-mkstandalone	Genererer et frittstående bilde
grub-ofpathname	Er et hjelpeprogram som skriver ut banen til en GRUBenhet
grub-probe	Undersøker enhetsinformasjon for en gitt bane eller enhet
grub-reboot	Angir standard oppstartsoppføring for GRUB bare for neste oppstart
grub-render-label	Gjengir Apple <code>.disk_label</code> for Apple Macer
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

8.65. Gzip-1.13

Gzip pakken inneholder programmer for komprimering og dekomprimering av filer.

Omtrentlig byggetid: 0.3 SBU

Nødvendig diskplass: 21 MB

8.65.1. Installasjon av Gzip

Forbered Gzip for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.65.2. 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 <code>.gz</code> filletternavn 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— <code>.z</code> til <code>.gz</code>

8.66. IPRoute2-6.13.0

IPRoute2 pakken inneholder programmer for grunnleggende og avansert IPV4 basert nettverksbygging.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 17 MB

8.66.1. 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 og en manside for **arpd** vil fortsatt bli installert. Forhindre dette ved å kjøre kommandoene nedenfor.

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

Kompiler pakken:

```
make NETNS_RUN_DIR=/run/netns
```

Denne pakken har ikke en fungerende testpakke.

Installer pakken:

```
make SBINDIR=/usr/sbin install
```

Hvis ønskelig, installer dokumentasjonen:

```
install -vDm644 COPYING README* -t /usr/share/doc/iproute2-6.13.0
```

8.66.2. Innhold i IPRoute2

Installerte programmer: bridge, ctstat (lenker til lstat), genl, ifstat, ip, lstat, nstat, routel, rtacct, rtmon, rtpr, rtstat (lenker til lstat), ss, og tc

Installerte mapper: /etc/iproute2, /usr/lib/tc, og /usr/share/doc/iproute2-6.13.0

Korte beskrivelser

bridge Konfigurerer nettverksbroer

ctstat Verktøy for tilkoblingsstatus

genl Generisk verktøy for netlink grenseflate

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 nabobindinger og deres egenskaper, legge til nye nabooppføringer og slette gamle
ip rule lar brukerne se på rutingsreglene og endre dem
ip route lar brukerne se på rutetabellen 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 lar brukere angi, endre eller slette multicast routing
ip monitor lar brukerne overvåke kontinuerlig tilstanden til enheter, adresser og ruter

lstat Gir Linux nettverksstatistikk; det er en generalisert og mer funksjonsfull erstatning for det gamle **rtstat** programmet

nstat	Viser nettverksstatistikk
routel	En komponent av ip route , for å liste rutetabellene
rtacct	Viser innholdet i <code>/proc/net/rt_acct</code>
rtmon	Overvåkingsverktøy for Route
rtpr	Konverterer utdataene til ip -o til en lesbar form
rtstat	Statusverktøy for Route
ss	Ligner på netstat kommandoen; viser aktive forbindelser
tc	Trafikkkontroll for Quality Of Service (QOS) og Class Of Service (COS) implementeringer tc qdisc lar brukere sette opp kødisiplinen tc class lar brukere sette opp klasser basert på køen til kødisiplinplanleggingen tc filter lar brukere sette opp QOS/COS pakkefiltrering tc monitor kan brukes til å se endringer gjort til trafikkkontroll i kjernen.

8.67. Kbd-2.7.1

Kbd pakken inneholder tastaturfiler, konsollfonter og tastaturverktøy.

Omtrentlig byggetid: 0.1 SBU

Nødvendig diskplass: 34 MB

8.67.1. Installasjon av Kbd

Oppførselen til tilbaketastene og slettetastene er ikke konsistent på tvers av tastaturopsettene i Kbd pakken. Følgende oppdatering fikser dette problemet for i386 tastaturopsett:

```
patch -Np1 -i ../kbd-2.7.1-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
```



Notat

For noen språk (f.eks. hviterussisk) gir ikke Kbd pakken et nyttig tastaturopsett hvor beholdningen «av» tastaturopsettene antar en ISO-8859-5-koding og CP1251 tastaturet vanligvis brukes. Brukere av slike språk må laste ned et fungerende tastaturopsett separat.

Hvis ønskelig, installer dokumentasjonen:

```
cp -R -v docs/doc -T /usr/share/doc/kbd-2.7.1
```

8.67.2. Innhold i Kbd

Installerte programmer: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (lenker til psfxtable), psfgettable (lenker til psfxtable), psfstriptime (lenker til 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-2.7.1, 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 tastaturkode (scancode-to-keycode) tilordningstabell
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 (unicode-to-font) kartleggingstabell
mapscrn	Et utdatert program som pleide å laste en brukerdefinert utdata 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 konsollfont
psfstrietable	Fjerner den innebygde Unicode tegntabellen fra en konsollfont
psfxtable	Håndterer Unicode tegntabeller for konsollfonter
setfont	Endrer fonter for forbedrede grafikkadapteren (EGA) og videografikk matrise (VGA) på konsollen
setkeycodes	Laster kjernens skanningskode til tastaturkode (scancode-to-keycode) kartleggingstabelloppføringer; dette er nyttig hvis det er uvanlige taster på tastaturet
setleds	Stiller inn tastaturflagg og lysdioder (LED)
setmetamode	Definerer tastaturets metatasthåndtering
setvtrgb	Stiller inn konsollfargekartet i alle virtuelle terminaler
showconsolefont	Viser gjeldende EGA/VGA konsollskjermfont
showkey	Rapporterer skanningskodene, tastekodene 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

8.68. Libpipeline-1.5.8

Libpipeline pakken inneholder et bibliotek for å manipulere kanaler av delprosesser på en fleksibel og praktisk måte.

Omtrentlig byggetid: 0.1 SBU

Nødvendig diskplass: 11 MB

8.68.1. Installasjon av Libpipeline

Forbered Libpipeline for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.68.2. Innhold i Libpipeline

Installert bibliotek: libpipeline.so

Korte beskrivelser

`libpipeline` Dette biblioteket brukes til å trygt konstruere kanaler mellom delprosesser

8.69. Make-4.4.1

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

Omtrentlig byggetid: 0.7 SBU

Nødvendig diskplass: 13 MB

8.69.1. Installasjon av Make

Forbered Make for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
chown -R tester .  
su tester -c "PATH=$PATH make check"
```

Installer pakken:

```
make install
```

8.69.2. 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

8.70. 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

8.70.1. Installasjon av patch

Forbered patch for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.70.2. Innhold i oppdateringen

Installert program: patch

Korte beskrivelser

patch Endrer filer i henhold til en patch fil (En patch fil er normalt en forskjellsoppløsing opprettet med **diff** programmet. Ved å bruke disse forskjellene på originalfilene, **patch** oppretter de lappede versjonene.)

8.71. Tar-1.35

Tar pakken gir muligheten til å lage tar arkiver og 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 er lagret.

Omtrentlig byggetid: 0.6 SBU
Nødvendig diskplass: 43 MB

8.71.1. 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 the `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 (på grunn av at LFS mangler `selinux`), men vil bli hoppet over hvis vertskjernen ikke støtter utvidede attributter eller sikkerhetsetiketter på filsystemet som brukes til å bygge LFS.

Installer pakken:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.35
```

8.71.2. Innhold i Tar

Installerte programmer: tar
Installert mappe: /usr/share/doc/tar-1.35

Korte beskrivelser

tar Oppretter, trekker ut filer fra og viser innholdet i arkiver, også kjent som tarballer

8.72. Texinfo-7.2

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

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 160 MB

8.72.1. Installasjon av Texinfo

Forbered Texinfo for kompilering:

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

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 tree 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
```

8.72.2. Innhold i Texinfo

Installerte programmer: info, install-info, makeinfo (lenker til texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf, og texindex

Installert bibliotek: MiscXS.so, Parsetexi.so, og XSParagraph.so (alle i `/usr/lib/texinfo`)

Installerte mapper: `/usr/share/texinfo` og `/usr/lib/texinfo`

Korte beskrivelser

info Brukes til å lese informasjonssider som ligner på mansider, 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** index file

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

8.73. Vim-9.1.1166

Vim pakken inneholder en kraftig tekstredigerer.

Omtrentlig byggetid: 3.4 SBU

Nødvendig diskplass: 251 MB



Alternativer til Vim

Hvis du foretrekker en annen tekstredigerer—som Emacs, Joe, eller Nano—Vennligst se <https://www.linuxfromscratch.org/blfs/view/12.3/postlfs/editors.html> for foreslåtte installasjonsinstruksjoner.

8.73.1. 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 og ekskluder én fil som inneholder tester som krever **curl** eller **wget**:

```
chown -R tester .
sed '/test_plugin_glvs/d' -i src/testdir/Make_all.mak
```

Kjør nå testene som bruker `tester`:

```
su tester -c "TERM=xterm-256color 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 (spesielt mens vi overstyrer `TERM` variabelen for å tilfredsstille noen forutsetninger for testpakken). 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 ferdigstilling.

Installer pakken:

```
make install
```

Mange brukere skriver refleksivt **vi** i stedet for **vim**. For å tillate kjøringen av **vim** når brukere vanligvis skriver **vi**, opprett en symbolkobling for både binærsiden og mandsiden 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-9.1.1166`, som gjør det i samsvar med plasseringen av dokumentasjonen for andre pakker:

```
ln -sv ../vim/vim91/doc /usr/share/doc/vim-9.1.1166
```

Hvis et X Window System skal installeres på LFS systemet, kan det være nødvendig å recompile 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/12.3/postlfs/vim.html>.

8.73.2. 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 vil 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 en vi kompatibel måte. Fjern «no» for å beholde det gamle **vi** oppførselen. *set backspace=2* innstillingen tillater tilbaketast over linjeskift, autoinnrykk og starten på et innlegg. *syntax on* parameteren 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** sin gjetting om bakgrunnsfargen til en eller annen terminalemulatorer. Dette gir uthevingen et bedre fargevalg for bruk på svart bakgrunn for disse programmene.

Dokumentasjon for andre tilgjengelige alternativer kan fås ved å kjøre følgende kommando:

```
vim -c ':options'
```



Notat

Som standard installerer vim kun stavefiler for det engelske språket. For å installere stavefiler for ditt foretrukne språk, kopier `.sp1` og eventuelt `.sug` filer for ditt språk og tegnkoding fra `runtime/spell` til mappen `/usr/share/vim/vim91/spell/`.

For å bruke disse stavefilene, trengs noen konfigurasjoner i `/etc/vimrc`, f.eks.:

```
set spelllang=en,ru
set spell
```

For mer informasjon, se `runtime/spell/README.txt`.

8.73.3. 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 ; ingen skallkommandoer kan startes og view kan ikke suspenderes
rvim	Er en begrenset versjon av vim ; ingen skallkommandoer 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 binære endringer

8.74. MarkupSafe-3.0.2

MarkupSafe er en Python modul som implementerer en XML/HTML/XHTML Markup sikker streng.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 500 KB

8.74.1. Installasjon av MarkupSafe

Kompiler MarkupSafe med følgende kommando:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
pip3 install --no-index --find-links dist Markupsafe
```

8.74.2. Innhold i MarkupSafe

Installert mappe: /usr/lib/python3.13/site-packages/MarkupSafe-3.0.2.dist-info

8.75. Jinja2-3.1.5

Jinja2 er en Pythonmodul som implementerer et enkelt pytonisk malspråk.

Omtrentlig byggetid: less than 0.1 SBU

Nødvendig diskplass: 2.5 MB

8.75.1. Installasjon av Jinja2

Bygg pakken:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Installer pakken:

```
pip3 install --no-index --find-links dist Jinja2
```

8.75.2. Innhold i Jinja2

Installerte mappe: /usr/lib/python3.13/site-packages/Jinja2-3.1.5.dist-info

8.76. Udev fra Systemd-257.3

Udev pakken inneholder programmer for dynamisk oppretting av enhetsnoder.

Omtrentlig byggetid: 0.3 SBU

Nødvendig diskplass: 161 MB

8.76.1. Installasjon av Udev

Udev er en del av systemd-257.3 pakken. Bruk systemd-257.3.tar.xz file som tarballkilde.

Fjern to unødvendige grupper, `render` og `sgx`, fra standard udev regler:

```
sed -e 's/GROUP="render"/GROUP="video"/' \
    -e 's/GROUP="sgx", //' \
    -i rules.d/50-udev-default.rules.in
```

Fjern en udevregel som krever en full Systemd installasjon:

```
sed -i '/systemd-sysctl/s/^/#/' rules.d/99-systemd.rules.in
```

Juster de hardkodede banene til nettverkskonfigurasjonsfiler for frittstående udev installasjon:

```
sed -e '/NETWORK_DIRS/s/systemd/udev/' \
    -i src/libsystemd/sd-network/network-util.h
```

Forbered Udev for kompilering:

```
mkdir -p build
cd      build

meson setup .. \
  --prefix=/usr \
  --buildtype=release \
  -D mode=release \
  -D dev-kvm-mode=0660 \
  -D link-udev-shared=false \
  -D logind=false \
  -D vconsole=false
```

Betydningen av mesonalternativene:

`--buildtype=release`

Denne bryteren overstyrer standard byggetype («debug»), som produserer uoptimaliserte binære filer.

`-D mode=release`

Deaktiver noen funksjoner som anses som eksperimentelle av oppstrøms.

`-D dev-kvm-mode=0660`

Standard udev regelen vil tillate alle brukere tilgang til `/dev/kvm`. Utviklerne anser det som farlig. Dette alternativet overstyrer det.

`-D link-udev-shared=false`

Dette alternativet forhindrer udev fra å koble til det interne systemd delte biblioteket, `libsystemd-shared`. Dette biblioteket er designet for å deles av mange Systemd komponenter og det er overdrevent for kun en udev installasjon.

`-D logind=false -Dvconsole=false`

Disse alternativene forhindrer generering av flere udev regelfiler som tilhører de andre Systemd komponentene som vi ikke vil installere.

Få listen over de sendte Udev hjelperne og lagre den i en miljøvariabel (eksportere den er strengt tatt ikke nødvendig, men den gjør det enklere å bygge som en vanlig bruker eller bruke en pakkebehandling):

```
export udev_helpers=$(grep "'name' :" ../src/udev/meson.build | \
    awk '{print $3}' | tr -d ",|" | grep -v 'udevadm')
```

Bygg bare komponentene som trengs for udev:

```
ninja udevadm systemd-hwdb \
    $(ninja -n | grep -Eo '(src/(lib)?udev|rules.d|hwdb.d)/[^ ]*') \
    $(realpath libudev.so --relative-to .) \
    $udev_helpers
```

Installerer pakken:

```
install -vm755 -d {/usr/lib,/etc}/udev/{hwdb.d,rules.d,network}
install -vm755 -d /usr/{lib,share}/pkgconfig
install -vm755 udevadm /usr/bin/
install -vm755 systemd-hwdb /usr/bin/udev-hwdb
ln -svfn ../bin/udevadm /usr/sbin/udev
cp -av libudev.so{,[0-9]} /usr/lib/
install -vm644 ../src/libudev/libudev.h /usr/include/
install -vm644 src/libudev/*.pc /usr/lib/pkgconfig/
install -vm644 src/udev/*.pc /usr/share/pkgconfig/
install -vm644 ../src/udev/udev.conf /etc/udev/
install -vm644 rules.d/* ../rules.d/README /usr/lib/udev/rules.d/
install -vm644 $(find ../rules.d/*.rules \
    -not -name '*power-switch*') /usr/lib/udev/rules.d/
install -vm644 hwdb.d/* ../hwdb.d/{*.hwdb,README} /usr/lib/udev/hwdb.d/
install -vm755 $udev_helpers /usr/lib/udev
install -vm644 ../network/99-default.link /usr/lib/udev/network
```

Installerer noen tilpassede regler og støttefiler som er nyttige i et LFS miljøet:

```
tar -xvf ../../udev-lfs-20230818.tar.xz
make -f udev-lfs-20230818/Makefile.lfs install
```

Installerer manualsiden:

```
tar -xf ../../systemd-man-pages-257.3.tar.xz \
    --no-same-owner --strip-components=1 \
    -C /usr/share/man --wildcards '*/udev*' '*/libudev*' \
    '*/systemd.link.5' \
    '*/systemd-'{hwdb,udev.service}.8

sed 's|systemd/network|udev/network|' \
    /usr/share/man/man5/systemd.link.5 \
    > /usr/share/man/man5/udev.link.5

sed 's/systemd(\\|\\|?-)udev\\1/' /usr/share/man/man8/systemd-hwdb.8 \
    > /usr/share/man/man8/udev-hwdb.8

sed 's|lib.*udev|sbin/udev|' \
    /usr/share/man/man8/systemd-udev.service.8 \
    > /usr/share/man/man8/udev.8

rm /usr/share/man/man*/systemd*
```

Til slutt, deaktiver `udev_helpers` variabelen:

```
unset udev_helpers
```

8.76.2. Konfigurerer Udev

Informasjon om maskinvareenheter opprettholdes i `/etc/udev/hwdb.d` og `/usr/lib/udev/hwdb.d` mappene. Udev trenger at informasjonen samles inn i en binær database `/etc/udev/hwdb.bin`. Opprett første database:

```
udev-hwdb update
```

Denne kommandoen må kjøres hver gang maskinvareinformasjonen er oppdatert.

8.76.3. Innhold i Udev

Installerte programmer: udevadm, udevd (symlenke til udevadm), og udev-hwdb
Installerte biblioteker: libudev.so
Installerte mapper: /etc/udev og /usr/lib/udev

Korte beskrivelser

udevadm Generisk udev administrasjonsverktøy: kontrollerer udevd nissen, gir informasjon fra Udevdatabasen, overvåker uevents, venter på uevents for å fullføre, tester Udevkonfigurasjonen og utløser uevents for en gitt enhet

udev En nisse som lytter etter uevents på netlink socket, oppretter enheter og kjører de konfigurerte eksterne programmene i respons på disse hendelsene

udev-hwdb Oppdaterer eller spørre maskinvaredatabasen.

libudev Et biblioteksgrensesnitt til udev enhetsinformasjon

/etc/udev Inneholder Udev konfigurasjonsfiler, enhetstillatelser og regler for navngivning av enheter

8.77. Man-DB-2.13.0

Man-DB pakken inneholder programmer for å finne og se på mandsider.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 44 MB

8.77.1. Installasjon av Man-DB

Forbered Man-DB for kompilering:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/man-db-2.13.0 \
            --sysconfdir=/etc \
            --disable-setuid \
            --enable-cache-owner=bin \
            --with-browser=/usr/bin/lynx \
            --with-vgrind=/usr/bin/vgrind \
            --with-grap=/usr/bin/grap \
            --with-systemdtmpfilesdir= \
            --with-systemdsystemunitdir=
```

Betydningen av konfigureringsalternativene:

`--disable-setuid`

Dette deaktiverer å lage **man** program setuid til bruker `man`.

`--enable-cache-owner=bin`

Dette endrer eierskapet til de systemomfattende hurtigbufferfilene til 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 mandsider. 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.

`--with-systemd...`

Disse parameterne forhindrer installasjon av unødvendig `systemd` mapper og filer.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.77.2. Ikke-engelske manualsider i LFS

Følgende tabell viser tegnsettet som Man-DB antar manuelle sider installert under `/usr/share/man/<11>` vil være kodet med. I tillegg til dette, bestemmer Man-DB korrekt om manualsider installert i den katalogen er UTF-8-kodet.

Tabell 8.1. Forventet tegnkode av eldre 8-biters manualsider

Språk (Kode)	Koding	Språk (Kode)	Koding
Dansk (da)	ISO-8859-1	Kroatisk (hr)	ISO-8859-2

Språk (Kode)	Koding	Språk (Kode)	Koding
Tysk (de)	ISO-8859-1	Ungarsk (hu)	ISO-8859-2
Engelsk (en)	ISO-8859-1	Japansk (ja)	EUC-JP
Spansk (es)	ISO-8859-1	Koreansk (ko)	EUC-KR
Estisk (et)	ISO-8859-1	Litauisk (lt)	ISO-8859-13
Finsk (fi)	ISO-8859-1	Latvisk (lv)	ISO-8859-13
Fransk (fr)	ISO-8859-1	Makedonsk (mk)	ISO-8859-5
Irsk (ga)	ISO-8859-1	Polsk (pl)	ISO-8859-2
Galisisk (gl)	ISO-8859-1	Rumensk (ro)	ISO-8859-2
Indonesisk (id)	ISO-8859-1	Gresk (el)	ISO-8859-7
Islandsk (is)	ISO-8859-1	Slovakisk (sk)	ISO-8859-2
Italiensk (it)	ISO-8859-1	Slovensk (sl)	ISO-8859-2
Norsk Bokmål (nb)	ISO-8859-1	Serbisk Latin (sr@latin)	ISO-8859-2
Nederlandsk (nl)	ISO-8859-1	Serbisk (sr)	ISO-8859-5
Norsk Nynorsk (nn)	ISO-8859-1	Tyrkisk (tr)	ISO-8859-9
Norsk (no)	ISO-8859-1	Ukrainsk (uk)	KOI8-U
Portugisisk (pt)	ISO-8859-1	Vietnamesisk (vi)	TCVN5712-1
Svensk (sv)	ISO-8859-1	Forenklet Kinesisk (zh_CN)	GBK
Hviterussisk (be)	CP1251	Forenklet Kinesisk, Singapore (zh_SG)	GBK
Bulgarsk (bg)	CP1251	Tradisjonell Kinesisk, Hong Kong (zh_HK)	BIG5HKSCS
tsjekkisk (cs)	ISO-8859-2	Tradisjonell Kinesisk (zh_TW)	BIG5



Notat

Manuallsider på språk som ikke er på listen støttes ikke.

8.77.3. 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-2.13.0

Korte beskrivelser

accessdb Dumper **whatis** databaseinnhold i menneskelig lesbar form

apropos Søker **whatis** databasen og viser de korte beskrivelsene av systemkommandoer som inneholder en gitt streng

catman Oppretter eller oppdaterer de forhåndsformaterte manualesidene

lexgrog Viser en-linjes sammendragsinformasjon om en gitt manualeside

man Formaterer og viser den forespurte manualesiden

man-recode Konverterer manualesider 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 separat ord
libman	Inneholder kjøretidsstøtte for man
libmandb	Inneholder kjøretidsstøtte for man

8.78. Procps-ng-4.0.5

Procps-ng pakken inneholder programmer for overvåking av prosesser.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 28 MB

8.78.1. Installasjon av Procps-ng

Forbered procps-ng for kompilering:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/procps-ng-4.0.5 \
            --disable-static \
            --disable-kill \
            --enable-watch8bit
```

Betydningen av konfigureringsalternativet:

`--disable-kill`

Denne bryteren deaktiverer bygging av **kill** kommandoen som vil bli installert av Util-linux pakken.

`--enable-watch8bit`

Denne bryteren aktiverer ncursesw støtte for **watch** kommandoen, slik at den kan håndtere 8-bit tegn.

Kompiler pakken:

```
make
```

For å kjøre testpakken, kjør:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

En test navngitt `ps with output flag bsdttime,cputime,etime,etimes` er kjent for å mislykkes hvis vertskjernen ikke er bygget med `CONFIG_BSD_PROCESS_ACCT` aktivert. I tillegg kan en `pgrep test` mislykkes i chroot miljøet.

Installer pakken:

```
make install
```

8.78.2. Innhold i Procps-ng

Installerte programmer: free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w, og watch
Installert bibliotek: libproc-2.so
Installerte mapper: /usr/include/procps og /usr/share/doc/procps-ng-4.0.5

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	Lister gjeldende prosesser
pwdx	Rapporterer gjeldende arbeidsmappe for en prosess
slabtop	Viser detaljert kjernens 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 som 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
<code>libproc-2</code>	Inneholder funksjonene som brukes av de fleste programmer i denne pakken

8.79. Util-linux-2.40.4

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

Omtrentlig byggetid: 0.5 SBU

Nødvendig diskplass: 316 MB

8.79.1. Installasjon av Util-linux

Forbered Util-linux for kompilering:

```
./configure --bindir=/usr/bin \
            --libdir=/usr/lib \
            --runstatedir=/run \
            --sbindir=/usr/sbin \
            --disable-chfn-chsh \
            --disable-login \
            --disable-nologin \
            --disable-su \
            --disable-setpriv \
            --disable-runuser \
            --disable-pylibmount \
            --disable-liblastlog2 \
            --disable-static \
            --without-python \
            --without-systemd \
            --without-systemdsystemunitdir \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.40.4
```

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, lag en dummy `/etc/fstab` fil å tilfredsstille to tester og kjør testpakken som en ikke-root bruker:



Advarsel

Å kjøre 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
```

```
touch /etc/fstab
chown -R tester .
su tester -c "make -k check"
```

`hardlink` tester vil mislykkes hvis vertens kjerne ikke har alternativet `CONFIG_CRYPTO_USER_API_HASH` aktivert eller har ingen alternativer som gir en SHA256 implementering (for eksempel, `CONFIG_CRYPTO_SHA256`, eller `CONFIG_CRYPTO_SHA256_SSSE3` hvis CPU støtter Supplemental SSE3) aktivert. I tillegg vil `lsfd: inotify`-testen mislykkes hvis kjernealternativet `CONFIG_NETLINK_DIAG` ikke er aktivert.

To andre tester, `lsfd: SOURCE column` og `utmp: last`, er kjent for å mislykkes i `chroot` miljøet.

Installer pakken:

```
make install
```

8.79.2. Innhold i Util-linux

Installerte programmer: addpart, agetty, blkdiscard, blkid, blkzone, blockdev, cal, cfdisk, chcpu, chmem, choom, chrt, col, colert, colrm, column, ctrlaltdel, delpart, dmesg, eject, fallocate, fdisk, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hardlink, hexdump, hwclock, i386 (lenker til setarch), ionice, ipcmk, ipcrm, ipcs, irqtop, isosize, kill, last, lastb (lenker til last), ldattach, linux32 (lenker til setarch), linux64 (lenker til setarch), logger, look, losetup, lsblk, lscpu, lsipc, lsirq, lsf, 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, rfcill, rtcwake, script, scriptlive, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff, swapon, switch_root, taskset, uclampset, ul, umount, uname26 (lenker til setarch), unshare, utmpdump, uidd, uuidgen, uuidparse, wall, wdctl, whereis, wipefs, x86_64 (lenker til setarch), 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-2.40.4, og /var/lib/hwclock

Korte beskrivelser

addpart	Informerer Linuxkjernen 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 blokkenhetsattributter
blkzone	Brukes til å administrere zone lagringsblokkenheter
blockdev	Lar brukere kalle blokkenhet ioctls fra kommandolinjen
cal	Viser en enkel kalender
cfdisk	Manipulerer partisjonstabellen til den gitte enheten
chcpu	Endrer tilstanden til CPUer
chmem	Konfigurerer minnet
choom	Viser og justerer OOM-killer poeng, brukes til å bestemme hvilken prosess som skal drepes først når Linux er tom for minne
chrt	Manipulerer sanntidsattributter til en prosess
col	Filtrerer ut omvendt linjemating
colert	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 Linuxkjernen om å fjerne en partisjon
dmesg	Dumper kjerneoppstartsmeldingene

eject	Løser ut flyttbare medier
fallocate	Forhåndsstiller plass til en fil
fdisk	Manipulerer partisjonstabellen til den gitte enheten
findcore	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	Setter en fillås og utfører deretter en kommando med låsen holdt
fsck	Brukes til å sjekke, og eventuelt reparere, filsystemer
fsck.cramfs	Utfører en konsistenssjekk på Cramfs filsystem på gitt enhet
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	Analysere alternativer i den gitte kommandolinjen
hardlink	Konsoliderer dupliserte filer ved å lage harde lenker
hexdump	Dumper den gitte filen i hexadecimal, decimal, octal, eller ascii
hwclock	Leser eller stiller inn systemets maskinvareklokke, også kalt sanntidsklokken (RTC) eller grunnleggende inndata-utdata system (BIOS) klokken
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 (Inter-Process Communication) ressursen
ipcs	Gir IPC statusinformasjon
irqtop	Viser informasjon om kjerneavbruddstiller i <i>top(1)</i> stilvisning
isozsize	Rapporterer størrelsen på et ISO9660 filsystem
kill	Sender signaler til prosesser
last	Viser hvilke brukere som sist logget på (og ut), søker tilbake gjennom <code>/var/log/wtmp</code> filen; den viser også systemoppstart, systemavslutning og endringer på kjørenivå
lastb	Viser mislykkede påloggingsforsøk, som logget i <code>/var/log/btmp</code>
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
lsfd	Viser informasjon om åpne filer; erstatter lsdf
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 områder 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 harddiskpartisjon)
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 stiene
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	Henter og angir ressursgrenser til en prosess
readprofile	Leser informasjon om kjerneprofileringen
rename	Gir 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
rfkill	Verktøy for å aktivere og deaktivere trådløse enheter
rtcwake	Brukes til å gå inn i systemets hviletilstand frem til den spesifiserte vekkingstiden
script	Lager et typeskript av en terminaløkt
scriptlive	Kjører 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 <code>root</code> å logge inn; den er normalt startet av init når systemet går i enkeltbrukermodus
swapon	Gjør endringer til vekselminneområdets 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 monteringsstreet
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
uuid	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 er et tilfeldig tall sannsynligvis unik blant alle UUID opprettet, på det lokale systemet og på andre systemer, i fortiden og i fremtiden, med ekstremt høy sannsynlighet (2^{128} UUIDs er mulig)
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 mansiden for den gitte kommandoen
wipefs	Sletter en filsystemsSignatur fra en enhet
x86_64	En symbolsk lenke til setarch
zramctl	Et program for å sette opp og kontrollere zram (komprimert ram disk) enheter
<code>libblkid</code>	Inneholder rutiner for enhetsidentifikasjon og symbol utdrag
<code>libfdisk</code>	Inneholder rutiner for manipulering av partisjonstabeller
<code>libmount</code>	Inneholder rutiner for montering og avmontering av en blokkenhet
<code>libsmartcols</code>	Inneholder rutiner for å hjelpe skjermtutdata i tabulatorform
<code>libuuid</code>	Inneholder rutiner for å generere unike identifikatorer for objekter som kan være tilgjengelig utenfor det lokale systemet

8.80. E2fsprogs-1.47.2

E2fsprogs pakken inneholder verktøyene for å håndtere `ext2` filsystemet. Det støtter også `ext3` og `ext4` journalførende filsystemer.

Omtrentlig byggetid: 2.4 SBU on a spinning disk, 0.5 SBU on an SSD
Nødvendig diskplass: 99 MB

8.80.1. 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-uidd     \
             --disable-fsck
```

Betydningen av konfigureringsalternativene:

`--enable-elf-shlibs`

Dette oppretter de delte bibliotekene som noen programmer i denne pakken bruker.

`--disable-*`

Dette hindrer E2fsprogs fra å bygge og installere `libuuid` og `libblkid` bibliotekene, `uidd` nissen, og `fsck` innpakningen, `util-linux` installerer nyere versjoner.

Kompiler pakken:

```
make
```

For å kjøre testene, utsted:

```
make check
```

En test navngitt `m_assume_storage_prezeroed` er kjent for å mislykkes. En annen test navngitt `m_rootdir_acl` er kjent for å mislykkes hvis filsystemet som brukes for LFS systemet ikke er `ext4`.

Installer pakken:

```
make install
```

Fjern ubrukelige statiske biblioteker:

```
rm -fv /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Denne pakken installerer en gzipped `.info` filen, men oppdaterer ikke den systemomfattende `dir` filen. Pakk ut denne filen og oppdater deretter systemets `dir` fil ved å bruke 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
```

8.80.2. Konfigurere E2fsprogs

`/etc/mke2fs.conf` inneholder standardverdien til ulike kommandolinjealternativer for **mke2fs**. Du kan redigere filen for å gjøre standardverdiene egnet for dine behov. For eksempel kan noen verktøy (ikke i LFS eller BLFS) ikke gjenkjenne en `ext4` filsystem med `metadata_csum_seed` funksjonen aktivert. **Hvis** du trenger et slikt verktøy, kan du fjerne funksjonen fra standard `ext4` funksjonsliste med kommandoen:

```
sed 's/metadata_csum_seed,/' -i /etc/mke2fs.conf
```

Les mansiden `mke2fs.conf(5)` for detaljer.

8.80.3. 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 mapper: /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 <code>ext{234}</code> filsystemer
compile_et	En feiltabellkompilator; den konverterer en tabell med navn på feilkoder og meldinger til en C kildefil som er egnet for bruk med <code>com_err</code> biblioteket
debugfs	En feilsøker for filsystemet; den kan brukes til å undersøke og endre tilstanden til <code>ext{234}</code> filsystemer
dumpe2fs	Skriver ut superblokken og gruppeinformasjon til blokker for filsystemet som finnes på en gitt enhet
e2freefrag	Rapporterer informasjon om fragmentering av ledig plass
e2fsck	Brukes til å sjekke, og eventuelt reparere <code>ext{234}</code> filsystemer
e2image	Brukes til å lagre kritiske <code>ext{234}</code> filsystemdata til en fil
e2label	Viser eller endrer filsystemetiketten på et <code>ext{234}</code> filsystem tilstede på en gitt enhet
e2mmpstatus	Sjekker MMP (Multiple Mount Protection) status for et <code>ext4</code> filsystem
e2scrub	Sjekker innholdet i et montert <code>ext{234}</code> filsystem
e2scrub_all	Sjekker alle monterte <code>ext{234}</code> filsystemer for feil
e2undo	Gjentar angreloggen (<code>undo_log</code>) for et <code>ext{234}</code> filsystem funnet på en enhet [Dette kan brukes til å angre en mislykket operasjon av et e2fsprogs program.]
e4crypt	<code>Ext4</code> filsystem krypteringsverktøy
e4defrag	Online defragmentering for <code>ext4</code> fil systemer
filefrag	Rapporter om hvor dårlig fragmentert en bestemt fil kan være
fsck.ext2	Som standard sjekker <code>ext2</code> filsystemer og er en hard lenke til e2fsck
fsck.ext3	Som standard sjekker <code>ext3</code> filsystemer og er en hard lenke til e2fsck
fsck.ext4	Som standard sjekker <code>ext4</code> filsystemer og er en hard lenke 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 <code>libss</code> delsystembibliotek
mke2fs	Oppretter et <code>ext{234}</code> filsystem på den angitte enheten
mkfs.ext2	Som standard oppretter <code>ext2</code> filsystemer og er en hard lenke til mke2fs
mkfs.ext3	Som standard oppretter <code>ext3</code> filsystemer og er en hard lenke til mke2fs
mkfs.ext4	Som standard oppretter <code>ext4</code> filsystemer og er en hard lenke til mke2fs
mklost+found	Oppretter en <code>lost+found</code> mappe på et <code>ext{234}</code> filsystem ; den forhåndstildeler diskblokker til denne mappen for å lette oppgaven til e2fsck
resize2fs	Kan brukes til å forstørre eller krympe et <code>ext{234}</code> filsystem
tune2fs	Justerer justerbare filsystemparametere på et <code>ext{234}</code> filsystem
<code>libcom_err</code>	Den vanlige feilvisningsrutinen
<code>libe2p</code>	Brukt av dumpe2fs , chattr , og lsattr
<code>libext2fs</code>	Inneholder rutiner for å gjøre det mulig for programmer på brukernivå å manipulere et <code>ext{234}</code> filsystem
<code>libss</code>	Brukt av debugfs

8.81. Sysklogd-2.7.0

Sysklogd pakken inneholder programmer for logging av systemmeldinger, slik som de kjernen gir når uvanlige ting skjer.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 4.1 MB

8.81.1. Installasjon av Sysklogd

Forbered pakken for kompilering:

```
./configure --prefix=/usr \
            --sysconfdir=/etc \
            --runstatedir=/run \
            --without-logger \
            --disable-static \
            --docdir=/usr/share/doc/sysklogd-2.7.0
```

Kompiler pakken:

```
make
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make install
```

8.81.2. Konfigurerer Sysklogd

Lag en ny `/etc/syslog.conf` fil ved å kjøre følgende:

```
cat > /etc/syslog.conf << "EOF"
# Begin /etc/syslog.conf

auth,authpriv.* -/var/log/auth.log
*.*;auth,authpriv.none -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *

# Do not open any internet ports.
secure_mode 2

# End /etc/syslog.conf
EOF
```

8.81.3. Innhold i Sysklogd

Installert program: syslogd

Korte beskrivelser

syslogd Logger meldingene for systemprogrammer som tilbyr logging [Hver logget melding inneholder minst et datostempel og et vertsnavn, og normalt programmets navn også men det avhenger av hvor tillitsfull loggingnissen blir fortalt å være.]

8.82. SysVinit-3.14

SysVinit pakken inneholder programmer for å kontrollere oppstarten, kjøring og avslutning av systemet.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 2.9 MB

8.82.1. Installasjon av SysVinit

Først bruker du en oppdatering som fjerner flere programmer installert av andre pakker, klargjør en melding og fikser en kompilatoradvarsel:

```
patch -Np1 -i ../sysvinit-3.14-consolidated-1.patch
```

Kompiler pakken:

```
make
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make install
```

8.82.2. Innhold i SysVinit

Installerte programmer: bootlogd, fstab-decode, halt, init, killall5, poweroff (lenker til halt), reboot (lenker til halt), runlevel, shutdown, og telinit (lenker til init)

Korte beskrivelser

bootlogd	Logger oppstartsmeldinger til en loggfil
fstab-decode	Kjører en kommando med fstab kodete argumenter
halt	Påkaller vanligvis shutdown med -h alternativet, men når du allerede er på kjørenivå 0, så ber den kjernen om å stoppe systemet; det noterer i filen <code>/var/log/wtmp</code> at systemet stenger ned
init	Den første prosessen som skal startes når kjernen har initialisert maskinvaren; den tar over oppstartsprosessen og starter alle prosesser spesifisert i konfigurasjonsfilen
killall5	Sender et signal til alle prosesser, bortsett fra prosessene i sin egen økt; den vil ikke drepe foreldreskallet
poweroff	Ber kjernen om å stoppe systemet og slå av datamaskinen (se halt)
reboot	Ber kjernen om å starte systemet på nytt (se halt)
runlevel	Rapporterer forrige og nåværende kjørenivå som nevnt i siste kjørenivå oppføring i <code>/run/utmp</code>
shutdown	Bringer systemet ned på en sikker måte, og signaliserer alle prosesser og varsler alle påloggede brukere
telinit	Forteller init hvilket kjørenivå den skal bytte til

8.83. Om feilsøkingssymboler

De fleste programmer og biblioteker er som standard kompilert med feilsøkingssymboler inkludert (med **gcc** sitt `-g` alternativ). Dette betyr at når du feilsøker et program eller bibliotek som ble kompilert med feilsøkingssymboler, kan feilsøkeren ikke bare gi minneadresser, men også navnene på rutinene og variablene.

Inkludering av disse feilsøkingssymbolene gjør imidlertid et program eller et bibliotek betydelig større. Følgende er et eksempel på hvor mye plass disse symbolene opptar:

- **bash** binær med feilsøkingssymboler: 1200 KB
- **bash** binær uten feilsøkingssymboler: 480 KB (60% mindre)
- Glibc og GCC filer (`/lib` og `/usr/lib`) med feilsøkingssymboler: 87 MB
- Glibc og GCC filer uten feilsøkingssymboler: 16 MB (82% mindre)

Størrelser vil variere avhengig av hvilken kompilator og C bibliotek som ble brukt, men et program som har blitt strippet for feilsøkingssymboler er vanligvis mellom 50 % til 80 % mindre enn dens ustrippede motpart. Fordi de fleste brukere aldri vil bruke en feilsøker på systemprogramvaren, kan mye diskplass gjenvinnes ved å fjerne disse symbolene. Den neste delen viser hvordan du fjerner alle feilsøkingssymboler fra programmene og bibliotekene.

8.84. 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, og noen unødvendige symboltabelloppføringer fra binærfiler og biblioteker. Dette medfører ingen ulemper for en typisk Linuxbruker.

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 ide å lage en sikkerhetskopiering av LFS systemet i gjeldende tilstand.

En **strip** kommando med `--strip-unnneeded` alternativet fjerner alle feilsøkingssymboler fra en binær eller et bibliotek. Og det fjerner alle symboltabelloppføringer som ikke er nødvendig for linkerens (for statiske biblioteker) eller dynamisk linker (for dynamisk koblede binære filer og delte biblioteker).

Feilsøkingssymbolene fra utvalgte biblioteker komprimeres med Zlib og lagres i separate filer. Denne feilsøkingssymbolinformasjonen er nødvendig for å kjøre regresjonstester med *valgrind* eller *gdb* senere i BLFS.

Merk 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 til `/tmp`, stripper dem der, og installer dem tilbake med **install** kommandoen. (Den relaterte oppføringen i Seksjon 8.2.1, «Oppgraderingsproblemer» gir begrunnelsen for å bruke **install** kommandoen her.)



Notat

ELF lasterens navn er `ld-linux-x86-64.so.2` på 64-bits systemer og `ld-linux.so.2` på 32-bits systemer. Konstruksjonen nedenfor velger riktig navn for gjeldende arkitektur, unntatt noe som slutter med `_g`, i tilfelle kommandoene nedenfor allerede har vært kjørt.



Viktig

Hvis det er en pakke som versjonen er forskjellig fra versjonen spesifisert av boken (enten etter et sikkerhetsråd eller som tilfredsstillende personlige preferanser), kan det være nødvendig å oppdatere bibliotekets filnavn i `save_usrlib` eller `online_usrbin`. **Unnlattelse av å gjøre det kan gjøre systemet helt ubrukelig.**

```
save_usrlib="$(cd /usr/lib; ls ld-linux*[^g])
            libc.so.6
            libthread_db.so.1
            libquadmath.so.0.0.0
            libstdc++.so.6.0.33
            libitm.so.1.0.0
            libatomic.so.1.2.0"

cd /usr/lib

for LIB in $save_usrlib; do
  objcopy --only-keep-debug $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-2.44.so
              libsframe.so.1.0.0
              libhistory.so.8.2
              libncursesw.so.6.5
              libm.so.6
              libreadline.so.8.2
              libz.so.1.3.1
              libzstd.so.1.5.7
              $(cd /usr/lib; find libnss*.so* -type f)"

for BIN in $online_usrbin; do
  cp /usr/bin/$BIN /tmp/$BIN
  strip --strip-unneeded /tmp/$BIN
  install -vm755 /tmp/$BIN /usr/bin
  rm /tmp/$BIN
done

for LIB in $online_usrlib; do
  cp /usr/lib/$LIB /tmp/$LIB
  strip --strip-unneeded /tmp/$LIB
  install -vm755 /tmp/$LIB /usr/lib
  rm /tmp/$LIB
done

for i in $(find /usr/lib -type f -name \*.so* ! -name \*dbg) \
         $(find /usr/lib -type f -name \*.a) \
         $(find /usr/{bin,sbin,libexec} -type f); do
  case "$online_usrbin $online_usrlib $save_usrlib" in
    *$(basename $i)* )
      ;;
    * ) strip --strip-unneeded $i
      ;;
  esac
done

unset BIN LIB save_usrlib online_usrbin online_usrlib
```

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.

8.85. Rydde opp

Til slutt, rydd 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". På et moderne Linuxsystem, er libtool .la-filene bare nyttig for libltdl. Ingen biblioteker i LFS forventes å bli lastet av libltdl, og det er kjent at noen .la-filer kan forårsake at BLFS pakker feiler under byggingen. Fjern disse filene nå:

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

For mer informasjon om libtool arkivfiler, se *BLFS delen "Om Libtool arkiverfiler (.la)"*.

Kompilatoren bygd i Kapittel 6 og Kapittel 7 er fortsatt delvis installert og ikke nldvendig 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
```

Kapittel 9. Systemkonfigurasjon

9.1. Introduksjon

Oppstart av et Linux system innebærer flere oppgaver. Prosessen må montere både virtuelle og ekte filsystemer, initialiser enheter, aktivere vekselfilen, sjekke filsystemer for integritet, montere eventuelle byttepartisjoner eller filer, sette systemklokken, få opp nettverk, start alle nisser (daemons) som kreves av systemet, og utføre alle andre tilpassede oppgaver brukeren trenger. Disse prosessene må organiseres for å sikre at oppgavene utføres på riktig måte og utføres så raskt som mulig.

9.1.1. System V

System V er den klassiske oppstartsprosessen som har blitt brukt i Unix og Unix lignende systemer som Linux siden ca 1983. Den består av et lite program, **init**, som setter opp grunnleggende programmer som f.eks **login** (via getty) og kjører et skript. Dette skriptet, vanligvis navngitt **rc**, kontrollerer utførelsen av et sett med ekstra skript som utfører oppgavene som kreves for å initialisere systemet.

init programmet er kontrollert av `/etc/inittab` filen og er organisert i kjørenivåer som kan kjøres av brukeren. I LFS brukes de som følgende:

```
0 — stopp
1 — Enkeltbrukermodus
2 — Brukerdefinerbar
3 — Full flerbrukermodus
4 — Brukerdefinerbar
5 — Full flerbrukermodus med skjermbehandler
6 — omstart
```

Vanlig standard kjørenivå er 3 eller 5.

Fordeler

- Etablert, godt forstått system.
- Enkelt å tilpasse.

Ulemper

- Kan være tregere å starte opp. Et middels hastighetsbasert LFS system tar 8-12 sekunder der oppstartstiden måles fra første kjernemelding til login ledeteksten. Nettverkstilkobling er vanligvis etablert ca. 2 sekunder etter påloggingsforespørselen.
- Seriell behandling av oppstartsoppgaver. Dette er relatert til forrige punkt. En forsinkelse i enhver prosess, for eksempel en filsystemsjekk, vil forsinke hele oppstartsprosessen.
- Støtter ikke direkte avanserte funksjoner som kontrollgrupper (cgroups) og planlegging av rettferdig andel per bruker.
- Å legge til skript krever manuelle, statiske sekvensbeslutninger.

9.2. LFS-Bootscripts-20240825

LFS-Bootscripts pakken inneholder et sett med skript for å starte/stoppe LFS systemet ved oppstart/avslutning. Konfigurasjonsfilene og prosedyrene som trengs for å tilpasse oppstartsprosessen er beskrevet i de følgende avsnittene.

Omtrentlig byggetid: less than 0.1 SBU
Nødvendig diskplass: 244 KB

9.2.1. Installasjon av LFS-Bootscripts

Installer pakken:

```
make install
```

9.2.2. Innhold i LFS-Bootscripts

Installerte skript: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountvirtfs, network, rc, reboot, sendsignals, setclock, ipv4-static, swap, sysctl, sysklogd, template, udev, og udev_retry
Installerte mapper: /etc/rc.d, /etc/init.d (symbolsk lenke), /etc/sysconfig, /lib/services, /lib/lsb (symbolsk lenke)

Korte beskrivelser

checkfs	Kontrollerer integriteten til filsystemene før de monteres (med unntak av journal- og nettverksbaserte filsystemer)
cleanfs	Fjerner filer som ikke skal bevares mellom omstart, for eksempel som de i /run/ og /var/lock/; det gjenskaper /run/utmp og fjerner de mulige tilstedeværende /etc/nologin, /fastboot, og /forcefsck filene
console	Laster inn riktig tastaturtabell for ønsket tastaturoppsett; den angir også skjermfonten
functions	Inneholder vanlige funksjoner, som feil- og statuskontroll, som brukes av flere bootscripts
halt	Stopper systemet
ifdown	Stopper en nettverksenhet
ifup	Initialiserer en nettverksenhet
localnet	Setter opp systemets vertsnavn og lokal tilbakekoblingsenhet
modules	Laster inn kjernemoduler som er oppflrt i /etc/sysconfig/modules, ved å bruke argumenter som også er gitt der
mountfs	Monterer alle filsystemer, bortsett fra de som er merket <i>noauto</i> , eller er nettverksbasert
mountvirtfs	Monterer virtuelle kjernefilssystemer, som f.eks <code>proc</code>
network	Setter opp nettverksgrensesnitt, for eksempel nettverkskort, og setter opp standard innfallsport (gateway) (der det er aktuelt)
rc	Hovedkontrollskriptet på kjørenivå; den er ansvarlig for å kjøre alle de andre bootsriptene en etter en, i en bestemt sekvens ved navnet på de symbolske lenkene som behandles
reboot	Starter systemet på nytt
sendsignals	Sørger for at hver prosess avsluttes før systemet starter på nytt eller stopper
setclock	Tilbakestiller kjernekløyken til lokal tid i tilfelle maskinvarerklokken ikke er satt til UTC
ipv4-static	Gir funksjonaliteten som trengs for å tildele et statisk Internett Protokoll (IP) adresse til et nettverksgrensesnitt

swap	Aktiverer og deaktiverer vekselfiler og partisjoner
sysctl	Laster systemkonfigurasjonsverdier fra <code>/etc/sysctl.conf</code> , hvis den filen eksisterer, inn i den kjørende kjernen
sysklogd	Starter og stopper system- og kjernelogg nissene (daemons)
template	En mal for å lage egendefinerte bootscripts for andre nisser
udev	Forbereder <code>/dev</code> mappen og starter Udev nissen
udev_retry	Forsøker på nytt mislykkede udev uevents, og kopier genererte regelfiler fra <code>/run/udev</code> til <code>/etc/udev/rules.d</code> hvis påkrevd

9.3. Oversikt over enhets- og modulhåndtering

I Kapittel 8, installerte vi udev nissen når udev 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 enhetsopprettingsmetode, hvor mange enhetsnoder ble opprettet under `/dev` (noen ganger bokstavelig talt tusenvis av noder), uavhengig av om de tilsvarende maskinvareenheter 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 kjernekernelen, 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. `devfs` ble merket som utdatert i lang tid, 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 det mulig å utvikle et brukerområde erstatning for `devfs`.

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 i `devtmpfs` filsystemet. Enhver driver som ønsker å registrere en enhetsnode vil bruke `devtmpfs` (via driverkjernen) for å gjøre det. Når en `devtmpfs` forekomst er montert på `/dev`, vil enhetsnoden i utgangspunktet bli eksponert for brukerrom med et fast navn, tillatelser og eieren.

Kort tid senere vil kjernen sende en uevent til **udev**. Basert på reglene spesifisert i filene i `/etc/udev/rules.d`, `/usr/lib/udev/rules.d`, og `/run/udev/rules.d` mappene, **udev** vil opprette flere symbolkoblinger til enhetsnoden, eller endre tillatelsene, eieren eller gruppen, eller endre den interne **udev** databaseoppføring (navn) for det objektet.

Reglene i disse tre mappene er nummererte og alle tre mappene slås sammen. Hvis **udev** ikke finner en regel for enheten den oppretter, vil den opprettholde tillatelsene og eierskapet som `devtmpfs` brukte i utgangspunktet.

9.3.2.3. Modullasting

Enhetsdrivere kompilert som moduler kan ha innebygde aliaser. Alias 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 gjøre at **udev** kaller `/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 lastning av uønskede drivere kan bli forhindre.

Kjernen selv er også i stand til å laste moduler for nettverksprotokoller, 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 uevent. Denne uevent håndteres deretter av **udev** 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-6.13.4, `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 enhetsmappen under `/sys/bus` og sjekk om det er en `modalias` fil 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` fil i den aktuelle mappen under `/sys/bus`, dette betyr at kjerneutviklerne ennå ikke har lagt til støtte for `modalias` for denne busstypen. Med Linux-6.13.4, dette er tilfellet med ISA busser. Forvent at dette problemet blir løst i senere kjerneversjoner.

`Udev` er ikke ment å laste «innpakkede (wrapper)» drivere som f.eks *snd-pcm-oss* og ikke-maskinvare drivere 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 den «innpakkede» modulen bare forbedrer 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** til å laste inn innpakningen etter at `udev` har lastet den innpakkede 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 (Før) `pre:` og (Etter) `post:` avhengigheter. Se *modprobe.d(5)* manualsiden for mer informasjon om «softdep» syntaks og muligheter.

Hvis den aktuelle modulen ikke er en innpakning og er nyttig i seg selv, konfigurer **modules** oppstartsskriptet til å laste denne modulen ved systemoppstart. For å gjøre dette, legg til modulnavnet i `/etc/sysconfig/modules` filen på en egen linje. Dette fungerer også for innpakkingsmoduler, men er ikke optimalt i så fall.

9.3.3.3. Udev laster inn noen uønskede moduler

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 regel 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

Først må du være sikker på at driveren er innebygd i kjernen eller allerede lastet inn som en modul, og at udev ikke oppretter en enhet med feil navn.

Hvis en kjernedriver ikke eksporterer dataene sine til `sysfs`, mangler udev informasjon som trengs for å opprette en enhetsnode. Dette vil mest sannsynlig skje med tredjepartsdrivere fra 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 at 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 Seksjon 9.4, «Administrere enheter» og Seksjon 9.5, «Generell nettverkskonfigurasjon» for eksempler.

9.3.4. Nyttig lesning

Ytterligere nyttig dokumentasjon er tilgjengelig på følgende nettsteder:

- En brukerromsimplesmentering av `devfs` http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf
- `sysfs` filsystemet <https://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf>

9.4. Administrere enheter

9.4.1. Nettverksenheter

Udev, som standard, navngir nettverksenheter i henhold til fastvare/BIOS data eller fysiske egenskaper som buss, spor eller MAC-adresse. Formålet med denne navnekonvensjonen er å sikre at nettverksenheter er navngitt konsekvent og ikke basert på tiden nettverkskortet var oppdaget. I eldre versjoner av Linux—på en datamaskin med to nettverkskort laget av Intel og Realtek, for eksempel—nettverkskort produsert av Intel kan ha blitt `eth0` mens Realtek-kortet ble `eth1`. Etter en omstart, er kortene noen ganger nummerert omvendt.

I det nye navneskjemaet vil typiske nettverksenhetsnavn være noe sånt som `enp5s0` eller `wlp3s0`. Hvis denne navnekonvensjonen ikke er ønsket, kan det tradisjonelle navneskjemaet eller et tilpasset oppsett være implementert.

9.4.1.1. Deaktivering av vedvarende navngivning på kjernekommandolinjen

Det tradisjonelle navneskjemaet som bruker `eth0`, `eth1`, osv. kan bli gjenopprettet ved å legge til `net.ifnames=0` på kjernens kommandolinje. Dette er mest passende for systemene som bare har en Ethernet-enhet av samme type. Bærbare datamaskiner har ofte flere ethernet-tilkoblinger som heter `eth0` og `wlan0` og er også kandidater for denne metoden. Kommandolinjen sendes i GRUB-konfigurasjonsfilen. Se Seksjon 10.4.4, «Opprette GRUB konfigurasjonsfilen»

9.4.1.2. Opprette tilpassede Udev regler

Navneskjemaet kan tilpasses ved å lage tilpassete udev regler. Et skript er inkludert som genererer de første reglene. Generer disse reglene ved å kjøre:

```
bash /usr/lib/udev/init-net-rules.sh
```

Nå, inspiser `/etc/udev/rules.d/70-persistent-net.rules` filen, for å finne ut hvilket navn som ble tildelt hvilken nettverksenhet:

```
cat /etc/udev/rules.d/70-persistent-net.rules
```



Notat

I noen tilfeller som når MAC-adresser har blitt tildelt et nettverkskort manuelt eller i et virtuelt miljø som Qemu eller Xen, blir kanskje ikke nettverksregelfilen generert fordi adresser ikke er konsekvent tildelt. I disse tilfellene kan denne metoden ikke bli brukt.

Filen begynner med en kommentarblokk etterfulgt av to linjer for hver NIC. Den første linjen for hvert NIC er en kommentert beskrivelse som viser dens maskinvare-IDer (f.eks. PCI leverandøren og enhets IDene, hvis det er et PCI-kort), sammen med driveren i parentes, hvis driveren kan bli funnet. Ingen maskinvare-IDer eller drivere brukes til å bestemme hvilket navn som skal gis grensesnittet; denne informasjonen er kun for referanse. Den andre linjen er udev regelen som samsvarer med dens NIC og faktisk tildeler det et navn.

Alle udev regler består av flere nøkkelord, atskilt med komma og valgfritt mellomrom. Her er nøkkelordene, og en forklaring på hvert enkelt:

- `SUBSYSTEM=="net"` - Dette forteller udev å ignorere enheter som ikke er nettverkskort.
- `ACTION=="add"` - Dette forteller udev å ignorere denne regelen for en hendelse som ikke er en `add` ("remove" og "change" hendelser), men trenger ikke å endre navn på nettverksgrensesnittene).
- `DRIVERS=="?*" - Dette eksisterer slik at udev vil ignorere VLAN- eller et bro undergrensesnitt (fordi disse undergrensesnittene ikke har drivere). Disse undergrensesnittene hopper over på grunn av navnet som ville bli tildelt ville kollidere med deres overordnede enheter.`
- `ATTR{address}` - Verdien av denne nøkkelen er NIC sin MAC adresse.
- `ATTR{type}=="1"` - Dette sikrer at regelen kun samsvarer med det primære grensesnittet for enkelte trådløse drivere som skaper flere virtuelle grensesnitt. De sekundære grensesnittene er hoppet over av samme grunn som VLAN og bro undergrensesnitt er hoppet over: ellers ville det vært en navnekollisjon.
- `NAME` - Verdien av denne nøkkelen er navnet som udev vil tilordne til dette grensesnittet.

Verdien av `NAME` er den viktige delen. Forsikre deg om at du vet hvilket navn som har blitt tildelt hvert av nettverkskortene dine før du fortsetter, og sørg for å bruke `NAME` verdien når du opprette konfigurasjonsfilene nedenfor.

Selv om den tilpassede UDEV regelfilen er opprettet, kan UDEV fortsatt tilordne ett eller flere alternative navn for en annen NIC basert på fysisk kjennetegn. Hvis en tilpasset UDEV regel ville gi nytt navn til en annen NIC ved hjelp av et navn som allerede er tildelt som et alternativt navn på en annen NIC, denne UDEV regelen vil mislykkes. Hvis dette problemet skjer, kan du opprette `/etc/udev/network/99-default.link` konfigurasjonsfil med en tom alternativ tildelingspolitikk, og overstyrer standard konfigurasjonsfil `/usr/lib/udev/network/99-default.link`:

```
sed -e '/^AlternativeNamesPolicy/s/=.*/=/' \
    /usr/lib/udev/network/99-default.link \
    > /etc/udev/network/99-default.link
```

9.4.2. CD-ROM symbolkoblinger

Noe programvare som du kanskje vil installere senere (f.eks. diverse mediaspillere) forventer at `/dev/cdrom` og `/dev/dvd` symbolkoblinger eksisterer, og å peke på en CD-ROM- eller DVD-ROM-enhet. Dessuten kan det være praktisk å sette referanser til disse symbolkoblingene i `/etc/fstab`. Udev kommer med et skript som vil generere regelfiler for å lage disse symbolkoblingene for deg, avhengig av egenskapene til hver enhet, men du må bestemme hvilken av to operasjonsmoduser du ønsker at skriptet skal bruke.

For det første kan skriptet operere i «by-path» modus (brukes som standard for USB- og FireWire-enheter), der reglene den oppretter avhenger av den fysiske banen til CD- eller DVD-enheten. For det andre kan den operere i «by-id» modus (standard for IDE- og SCSI-enheter), de reglene den oppretter avhenger av identifikasjonsstrenger som er lagret på selve CD eller DVD enheten. Banen bestemmes av udev sitt `path_id` skript, og identifikasjonsstregene leses fra maskinvaren av `ata_id` eller `scsi_id` programmene, avhengig av hvilken type enhet du har.

Det er fordeler med hver tilnærming; riktig tilnærming til bruk vil avhenge av hva slags enhetsendringer som kan skje. Hvis du forventer at fysisk vei til enheten (det vil si portene og/eller sporene som den kobler til) endres, for eksempel fordi du planlegger å flytte stasjonen til en annen IDE-port eller en annen USB-kontakt, så bør du bruke «by-id»modus. På den annen side, hvis du forventer at enhetens identifikasjon endres, for eksempel fordi det kan dø, og du vil erstatte den med en annen enhet med de samme egenskapene og som er koblet til de samme kontaktene, bør du bruke «by-path» modus.

Hvis begge endringene er mulig med stasjonen din, velg en modus basert på typen endring du forventer skal skje oftest.



Viktig

Eksterne enheter (for eksempel en USB-tilkoblet CD-stasjon) bør ikke vedvarende bruke by-path, fordi hver gang enheten kobles til en ny ekstern port, vil dens fysiske bane endres. Alle eksternt tilkoblede enheter vil ha dette problemet hvis du skriver udev-regler som gjenkjenner dem på deres fysiske sti; problemet er ikke begrenset til CD og DVD-stasjoner.

Hvis du ønsker å se verdiene som udev-skriptene vil bruke, for den aktuelle CD-ROM-enheten, finn den tilsvarende mappen under `/sys` (f.eks. kan dette være `/sys/block/hdd`) og kjør en kommando som ligner på følgende:

```
udevadm test /sys/block/hdd
```

Se på linjene som inneholder utdata fra forskjellige `*_id` programmer. «by-id» modus vil bruke `ID_SERIAL` verdien hvis den finnes og ikke er tom, ellers vil den bruke en kombinasjon av `ID_MODEL` og `ID_REVISJON`. «by-path» modus vil bruke `ID_PATH` verdien.

Hvis standardmodusen ikke passer for din situasjon, så kan følgende modifikasjoner gjøres på `/etc/udev/rules.d/83-cdrom-symlinks.rules` filen, som følgende (hvor `mode` er en av «by-id» eller «by-path»):

```
sed -e 's/"write_cd_rules"/"write_cd_rules mode"/' \
    -i /etc/udev/rules.d/83-cdrom-symlinks.rules
```

Merk at det ikke er nødvendig å lage regelfilene eller symbolkoblingene på dette tidspunktet fordi du har bind-montert vertens `/dev` mappe inn i LFS systemet og vi antar at symbolkoblingene finnes på verten. Reglene og symbolkoblingene vil opprettes første gang du starter opp LFS systemet.

Men hvis du har flere CD-ROM-enheter, kan symbolkoblingene generert på det tidspunktet peke på andre enheter enn de peker på hos verten din fordi enheter ikke oppdages i en forutsigbar rekkefølge. Den første tildelingen opprettet når du starter opp LFS systemet vil være stabile, så dette er bare et problem hvis du trenger symbolkoblingene på begge systemene til å peke på samme enhet. Hvis du trenger det, inspiser (og muligens rediger) den genererte `/etc/udev/rules.d/70-persistent-cd.rules` filen etter oppstart, for å sikre at de tilordnede symbolkoblingene samsvarer med det du trenger.

9.4.3. Håndtere dupliserte enheter

Som forklart i Seksjon 9.3, «Oversikt over enhets- og modulhåndtering» rekkefølgen som 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 Seksjon 9.5, «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+="webcam"
KERNEL=="video*", ATTRS{device}=="0x036f", ATTRS{vendor}=="0x109e", SYMLINK+="tvvtuner"

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/tvvtuner` og `/dev/webcam` som alltid peker på den rette enheten.

9.5. Generell nettverkskonfigurasjon

9.5.1. Opprette konfigurasjonsfiler for nettverksgrensesnitt

Hvilke grensesnitt bringes opp og ned av nettverksskriptet avhenger vanligvis av filene i `/etc/sysconfig/`. Denne mappen bør inneholde en fil for hvert grensesnitt som skal konfigureres, for eksempel `ifconfig.xyz`, hvor «xyz» skal beskrive nettverkskortet. Grensesnittnavnet (f.eks. `eth0`) er vanligvis passende. Inne i denne filen er attributter til dette grensesnittet, for eksempel dets IP adresse(r), nettverksmasker og så videre. Det er nødvendig at stammen av filnavnet er `ifconfig`.



Notat

Hvis prosedyren i forrige avsnitt ikke ble brukt, udev vil tildele nettverkskortgrensesnittnavn basert på systemets fysiske egenskaper som `enp2s1`. Hvis du ikke er sikker på hva grensesnittnavnet ditt er, kan du alltid kjøre **ip link** eller **ls /sys/class/net** etter at du har startet opp systemet.

Grensesnittnavnene avhenger av implementeringen og konfigurasjon av udev nissen (daemon) som kjører på systemet. udev nissen for LFS (installert i Seksjon 8.76, «Udev fra Systemd-257.3») vil ikke kjøre før LFS systemet er startet opp. Så det er upålitelig å bestemme grensesnittnavnene som brukes i LFS systemet ved å kjøre disse kommandoene på vertsdistroen, *selv om det er i chroot miljøet*.

Følgende kommando oppretter en eksempelfil for *eth0* enheten med en statisk IP-adresse:

```
cd /etc/sysconfig/
cat > ifconfig.eth0 << "EOF"
ONBOOT=yes
IFACE=eth0
SERVICE=ipv4-static
IP=192.168.1.2
GATEWAY=192.168.1.1
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

Verdiene i kursiv må endres i hver fil for å samsvare med riktig oppsett.

Hvis `ONBOOT` variabelen er satt til `yes` System V nettverksskript vil hente opp nettverksgrensesnittkortet (NIC) under systemets oppstartsprosess. Hvis satt til noe annet enn `yes` NIC vil bli ignorert av nettverksskriptet og ikke automatisk hentet frem. Grensesnittet kan startes eller stoppes manuelt med **ifup** og **ifdown** kommandoene.

`IFACE` variabelen definerer grensesnittnavnet, for eksempel `eth0`. Det kreves for all konfigurasjon av nettverksenhets filer. Filnavnet må samsvare med denne verdien.

`SERVICE` variabelen definerer metoden som brukes for å få IP adressen. LFS-Bootscripts pakken har en modulær IP tildelingsformat, og oppretter flere filer i `/lib/services/` katalogen tillater annen IP tildelingsmetoder. Dette brukes ofte for dynamisk vertskonfigurasjon Protokoll (DHCP), som er adressert i BLFS boken.

`GATEWAY` variabelen skal inneholde standard standardruter (gateway) IP adresse, hvis en er til stede. Hvis ikke, så kommenter variabelen helt ut.

`PREFIX` variabelen inneholder antall biter som brukes i subnett. Hver oktett i en IP-adresse er 8 bits. Hvis subnettets nettmasker er `255.255.255.0`, bruker den deretter de tre første oktettene (24 bits) for å spesifisere nettverksnummeret. Hvis nettmasken er `255.255.255.240`, ville det være å bruke de første 28 bitene. Prefikser lengre enn 24 biter er ofte brukt av DSL- og kabelbaserte Internett-leverandører (ISP). I dette eksemplet (`PREFIX=24`) er nettmasken `255.255.255.0`. Juster `PREFIX` variabel i henhold til ditt spesifikke undernett. Hvis det utelates, er `PREFIX` sin standard 24.

For mer informasjon se **ifup** man side.

9.5.2. Opprette `/etc/resolv.conf` filen

Systemet vil trenge noen midler for å få domenenavntjeneste (DNS) navneoppløsning for å løse Internett domenenavn til IP adresser, og omvendt. Dette oppnås best ved å plassere IP adressen til DNS server, tilgjengelig fra Internett leverandøren eller nettverksadministratoren, til `/etc/resolv.conf`. Opprett filen ved å kjøre følgende:

```
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 mansiden for `resolv.conf` for flere detaljer.

Erstatt `<IP adressen til din navneserver>` med IP adressen til DNS som passer best for oppsettet. Det vil ofte være mer enn en oppføring (krever sekundære servere for reservefunksjon). Hvis du bare trenger eller vil ha en DNS-server, fjern andre `nameserver` linjen fra filen. IP adressen kan også være en ruter på det lokale nettverket.



Notat

Googles offentlige IPv4 DNS adresser er `8.8.8.8` og `8.8.4.4`.

9.5.3. 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 "<ifs>" > /etc/hostname
```

`<ifs>` må erstattes med navnet til datamaskinen. Ikke skriv inn det fullt kvalifiserte domenenavnet (FQDN) her. Den informasjonen er i `/etc/hosts` filen.

9.5.4. Tilpassing av `/etc/hosts` filen

Bestem deg for et fullt kvalifisert domenenavn (FQDN) og mulige aliaser til bruk i `/etc/hosts` filen. Hvis du bruker statiske IP adresser, må du også bestemme deg for en IP adresse. Syntaksen for en vertsfiloppføring er:

```
IP_adresse 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 192.168.1.2.

Hvis datamaskinen skal være synlig for Internett, en gyldig FQDN kan være selve domenenavnet, eller en streng som er resultatet av å sette sammen et prefiks (ofte vertsnavnet) og domenenavnet med «.» karakter. Og du må kontakte domeneleverandøren for å knytte FQDN til din offentlige IP-adresse.

Selv om datamaskinen ikke er synlig for Internett, er en FQDN fortsatt nødvendig for at visse programmer, for eksempel MTA-er, skal fungere ordentlig. En spesiell FQDN, `localhost.localdomain`, kan bli brukt til dette formålet.

Opprett `/etc/hosts` filen ved å kjøre:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.1.2> <FQDN> <HOSTNAME> [alias1] [alias2 ...]
::1      localhost ip6-localhost ip6-loopback
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

# End /etc/hosts
EOF
```

`<192.168.1.2>`, `<FQDN>`, og `<HOSTNAME>` verdiene må være endret for spesifikke bruksområder eller krav (hvis tildelt en IP-adresse av en nettverks-/systemadministrator og maskinen kobles til en eksisterende nettverk). De valgfrie aliasnavnene kan utelates.

9.6. System V Oppstartskript Bruk og Konfigurasjon

9.6.1. Hvordan fungerer System V Oppstartskriptet?

Denne versjonen av LFS bruker en spesiell oppstartsfunksjon kalt SysVinit, basert på en serie av *kjørenivåer*. Det kan være ganske forskjellig fra et system til et annet, så det kan ikke antas at fordi ting fungerte i en spesiell Linux distribusjon, bør de fungere på samme måte i LFS også LFS har sin egen måte å gjøre ting på, men den respekterer allment aksepterte standarder.

Det er en alternativ oppstartsprosedyre kalt **systemd**. Vi vil ikke diskutere den oppstartsprosessen mer her. For en detaljert beskrivelse besøk <https://www.linux.com/training-tutorials/understanding-and-using-systemd/>.

SysVinit (som vil bli referert til som «init» fra nå av) fungerer ved å bruke en kjørenivåordning. Det er syv (nummerert 0 til 6) kjørenivåer (faktisk er det flere kjørenivåer, men de er for spesielle tilfeller og er vanligvis ikke brukt. Se *init(8)* for flere detaljer). Hver av disse tilsvarer handlingene datamaskinen skal utføre når den starter opp. Standard kjørenivå er 3. Her er beskrivelser av de ulike kjørenivåene etter hvert som de implementeres i LFS:

- 0: stopper datamaskinen
- 1: enkeltbrukermodus
- 2: reservert for tilpasning, ellers gjøres det samme som 3
- 3: flerbrukermodus med nettverk
- 4: reservert for tilpasning, ellers gjør den det samme som 3
- 5: samme som 4, den brukes vanligvis for GUI pålogging (som GNOME sin **gdm** eller LXDE sin **lxdm**)
- 6: starter datamaskinen på nytt



Notat

Klassisk ble kjørenivå 2 ovenfor definert som «flerbrukermodus uten nettverk», men dette var bare tilfelle for mange år siden da flere brukere kunne logge på et system koblet via serielle porter. I dagens miljø gir det ingen mening og vi betegner det nå som «reserved».

9.6.2. Konfigurere Sysvinit

Under kjerneinitialiseringen, det første programmet som kjøres (hvis ikke overstyrt på kommandolinjen) er **init**. Dette programmet leser initialiseringsfilen `/etc/inittab`. Opprette denne filen med:

```
cat > /etc/inittab << "EOF"
# Begin /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc S

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

ca:l2345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S06:once:/sbin/sulogin
sl:l:respawn:/sbin/sulogin

1:2345:respawn:/sbin/agetty --noclear tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab
EOF
```

En forklaring av denne initialiseringsfilen er på mansiden for *inittab*. I LFS, nøkkelkommandoen er **rc**. Initialiseringsfilen ovenfor instruerer **rc** å kjøre alle skriptene som starter med en **S** i `/etc/rc.d/rcS.d` mappen etterfulgt av alle skriptene som begynner med en **S** i `/etc/rc.d/rc?.d` mappen hvor spørsmålsteget er spesifisert av `initdefault` verdien.

Som en bekvemmelighet **rc** skriptet leser et bibliotek av funksjoner i `/lib/lsb/init-functions`. Dette biblioteket leser også en valgfri konfigurasjonsfil, `/etc/sysconfig/rc.site`. Hvilket som helst av systemets konfigurasjonsfilparametere beskrevet i påfølgende avsnitt kan være alternativt plassert i denne filen som tillater konsolidering av alle systemets parametere i denne ene filen.

Som en bekvemmelighet for feilsøking logger funksjonsskriptet også all utdata til `/run/var/bootlog`. Siden `/run` mappen er en tmpfs, er denne filen ikke vedvarende på tvers av oppstarter; men den er lagt til den mer permanente filen `/var/log/boot.log` på slutten av oppstartsprosessen.

9.6.2.1. Endre kjørenivåer

Endring av kjørenivåer er gjort med **init** `<runlevel>`, hvor `<runlevel>` er målet for kjørenivå. For eksempel for å starte datamaskinen på nytt, kan en bruker utstede **init 6** kommandoen, som er et alias for **reboot** kommandoen. Likeså er, **init 0** et alias for **halt** kommandoen.

Det er en rekke mapper under `/etc/rc.d` som ser ut som `rc?.d` (hvor `?` er nummeret på kjørenivået) og `rcS.d`, alle inneholder en rekke symbolske lenker. Noen begynner med en **K**; de andre begynner med en **S**, og alle av dem har to tall etter forbokstaven. **K** betyr å stoppe (drepe) en tjeneste og **S** betyr å starte en tjeneste. Tallene bestemmer rekkefølgen skriptene kjøres i, fra 00 til 99—jo lavere tall, desto tidligere blir det utført. Når **init** bytter til et annet kjørenivå de aktuelle tjenestene er enten startet eller stoppet, avhengig av valgt kjørenivå.

De virkelige skriptene er i `/etc/rc.d/init.d`. De gjør selve jobben, og alle symbolkoblingene peker til dem. K-lenker og S-lenker peker på samme skript i `/etc/rc.d/init.d`. Dette er fordi skriptene kan kalles med forskjellige parametere som `start`, `stop`, `restart`, `reload`, og `status`. Når en K-lenke oppstår, det aktuelle skriptet kjøres med `stop` argumentet. Når en S-kobling oppstår, kjøres det riktige skriptet med `start` argumentet.

Dette er beskrivelser av hva argumentene får skriptene til å gjøre:

`start`

Tjenesten blir startet.

`stop`

Tjenesten blir stoppet.

`restart`

Tjenesten stoppes og startes deretter på nytt.

`reload`

Konfigurasjonen av tjenesten blir oppdatert. Dette brukes etter at konfigurasjonsfilen til en tjeneste ble endret, når tjenesten ikke trenger å startes på nytt.

`status`

Forteller om tjenesten kjører og med hvilken PID.

Du kan gjerne endre måten oppstartsprosessen fungerer på (tross alt, det er ditt eget LFSsystem). Filene gitt her er et eksempel på hvordan det kan gjøres.

9.6.3. Udev oppstartsskripter

`/etc/rc.d/init.d/udev` initskriptet starter **udev**, trigger alle "coldplug"-enheter som allerede er opprettet av kjernen og venter på at eventuelle regler skal fullføres. Skriptet fjerner også uevent behandleren fra standarden på `/sbin/hotplug`. Dette gjøres fordi kjernen ikke lenger trenger å kalle en ekstern binær. I stedet vil **udev** lytte på en netlink socket for uevents som kjernen løfter.

`/etc/rc.d/init.d/udev_retry` initskriptet tar kontroll for å utløse hendelser på nytt for undersystemer som reglene kan avhenge av filsystemer som ikke er montert før **mountfs** skriptet kjøres (spesielt, `/usr` og `/var` kan forårsake dette). Disse skriptene kjører etter **mountfs** skriptet, så de reglene (hvis de utløses på nytt) bør lykkes andre gang. De er konfigurert fra `/etc/sysconfig/udev_retry` filen; alle andre ord i denne filen enn kommentarer regnes som undersystemnavn for å utløses ved nytt forsøk. For å finne delsystemet til en enhet, bruk **udevadm info --attribute-walk <device>** hvor `<device>` er en absolutt sti i `/dev` eller `/sys` som f.eks `/dev/sr0` eller `/sys/class/rtd`.

For informasjon om lasting av kjernemoduler og udev, se Seksjon 9.3.2.3, «Modullasting»

9.6.4. Konfigurering av systemklokken

setclock skriptet leser tiden fra maskinvareklokken, også kjent som BIOS eller Complementary Metal Oxide Semiconductor (CMOS) klokke. Hvis maskinvareklokken er satt til UTC, vil dette skriptet konvertere maskinvareklokkens tid til lokal tid ved å bruke `/etc/localtime` filen (som forteller **hwclock** programmet hvilken tidssone som skal brukes). Det er ingen måte å oppdage om maskinvareklokken er satt til UTC eller ikke, så dette må konfigureres manuelt.

setclock programmet kjøres via udev når kjernen oppdager maskinvare kapasiteten ved oppstart. Den kan også kjøres manuelt med `stop` parameteren til å lagre systemtiden til CMOS-klokken.

Hvis du ikke husker 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 maskinvare klokken. Hvis denne tiden samsvarer med hva klokken din sier, er maskinvareklokken satt til lokal tid. Hvis utdataen fra **hwclock** ikke er lokal tid, sjansen er 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 din lokale tid.

Endre verdien på `UTC` variabelen nedenfor til verdien `0` (null) hvis maskinvareklokken **IKKE** er satt til UTC-tid.

Opprett en ny fil `/etc/sysconfig/clock` ved å kjøre følgende:

```
cat > /etc/sysconfig/clock << "EOF"
# Begin /etc/sysconfig/clock

UTC=1

# Set this to any options you might need to give to hwclock,
# such as machine hardware clock type for Alphas.
CLOCKPARAMS=

# End /etc/sysconfig/clock
EOF
```

Et godt hint som forklarer hvordan man skal håndtere tid på LFS er tilgjengelig på <https://www.linuxfromscratch.org/hints/downloads/files/time.txt>. Den forklarer problemstillinger som f.eks tidssoner, UTC og `TZ` miljøvariabelen.



Notat

Parameterne `CLOCKPARAMS` og `UTC` kan også angis i `/etc/sysconfig/rc.site` filen.

9.6.5. Konfigurering av Linux konsollen

Denne delen diskuterer hvordan du konfigurerer **console** oppstartsskriptet som setter opp tastaturkartet, konsollfonten og konsollkjerneloggens nivå. Hvis ikke-ASCII-tegn (f.eks. copyright-tegnet, britiske pund tegn og eurosymbol) ikke vil bli brukt, og tastaturet er et amerikansk, kan mye av denne delen hoppes over. Uten konfigurasjonsfilen, (eller tilsvarende innstillinger i `rc.site`), **console** vil ikke oppstartsskriptet gjøre noe.

console skriptet leser `/etc/sysconfig/console` filen for konfigurasjons informasjon. Bestem hvilket tastatur og skjermkrift som skal brukes. Diverse språkspesifikke HOWTOer kan også hjelpe med dette, se <https://tldp.org/HOWTO/HOWTO-INDEX/other-lang.html>. Hvis du fortsatt er i tvil, se i `/usr/share/keymaps` og `/usr/share/consolefonts` mappene for gyldige tastaturer og skjermfonter. Les *loadkeys(1)* and *setfont(8)* manualsider for å finne de riktige argumenter for disse programmene.

`/etc/sysconfig/console` filen skal inneholde linjer av formen: `VARIABLE=verdi`. Følgende variabler gjenkjennes:

LOGLEVEL

Denne variabelen spesifiserer loggnivået for kjernemeldinger som sendes til konsollen som angitt av **dmmsg -n**. Gyldige nivåer er fra 1 (ingen beskjeder) til 8. Standardnivået er 7, som er ganske detaljert.

KEYMAP

Denne variabelen spesifiserer argumentene for **loadkeys** programmet, vanligvis navnet på tastaturet å laste, f.eks., `it`. Hvis denne variabelen ikke er satt, vil oppstartsskriptet ikke kjøre **loadkeys** programmet, og standard kjernetastatur vil bli brukt. Merk at noen få tastaturer har flere versjoner med samme navn (`cz` og dens varianter i `qwerty/` og `qwertz/`, `es` i `olpc/` og `qwerty/`, og `trf` i `fgGIod/` og `qwerty/`). I disse tilfellene bør også overordnet mappe spesifiseres (f.eks. `qwerty/es`) for å sikre at det riktige tastaturet er lastet.

KEYMAP_CORRECTIONS

Denne (sjelden brukte) variabelen spesifiserer argumentene for det andre kallet til **loadkeys** programmet. Dette er nyttig hvis keymap ikke helt er tilfredsstillende og en liten justering må gjøres. f.eks. å inkludere eurotegnet i et tastatur som vanligvis ikke har det, sett denne variabelen til `euro2`.

FONT

Denne variabelen spesifiserer argumentene for **setfont** programmet. Vanligvis inkluderer dette fontnavnet, `-m`, og navnet på programtegnkartet som skal lastes. For eksempel for å laste inn «lat1-16» fonten sammen med «8859-1» applikasjonskartet (som er det passende i USA), sett denne variabelen til `lat1-16 -m 8859-1`. I UTF-8 modus bruker kjernen applikasjonens tegnkart for konvertering av sammensatte 8-bits nøkkeltaster i keymap til UTF-8, og dermed argumentet til parameteren `-m` bør settes til kodingen av komponerte tastaturkoder i tastaturkartet.

UNICODE

Sett denne variabelen til `1`, `yes`, eller `true` for å sette konsollen til UTF-8 modus. Dette er nyttig i UTF-8 baserte lokaliteter og skadelig ellers.

LEGACY_CHARSET

For mange tastaturoppsett er det ikke noe Unicode-tastatur i Kbd pakken. **console** oppstartsskriptet vil konvertere et tilgjengelig tastaturkartet til UTF-8 umiddelbart hvis denne variabelen er satt til kodingen av det tilgjengelige tastaturet som ikke er UTF-8.

Noen eksempler:

- Vi vil bruke `C.UTF-8` som lokalitet for interaktive økter i Linuxkonsollen i Seksjon 9.7, «Konfigurere systemlokaliteten» så vi bør sette `UNICODE` til `1`. Konsollfontene levert av Kbd pakken som inneholder glyfer for alle tegnene fra programmeldingene i `C.UTF-8` lokalitet er `LatArCyrHeb*.psfu.gz`, `LatGrkCyr*.psfu.gz`, `Lat2-Terminus16.psfu.gz`, og `pancyrillic.f16.psfu.gz` i `/usr/share/consolefonts` (de andre leverte konsollfonter mangler glyffer av noen tegn som Unicode venstre/høyre anførselstegn og Unicode engelsk bindestrek). Så sett en av dem, for eksempel `Lat2-Terminus16.psfu.gz` som standard konsollfont:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
FONT="Lat2-Terminus16"

# End /etc/sysconfig/console
EOF
```

- For et ikke-Unicode oppsett er bare `KEYMAP`- og `FONT`-variablene generelt nødvendige. For eksempel for et polsk oppsett, ville man bruke:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# End /etc/sysconfig/console
EOF
```

- Som nevnt ovenfor er det noen ganger nødvendig å justere et tastaturoppsett litt. Følgende eksempel legger til Euro-symbolet til Tysk tastaturkart:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"
UNICODE="1"

# End /etc/sysconfig/console
EOF
```

- Følgende er et Unicode-aktivert eksempel for bulgarsk, hvor en UTF-8 keymap finnes:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# End /etc/sysconfig/console
EOF
```

- På grunn av bruken av en 512-glyph LatArCyrHeb-16 font i forrige eksempel, lyse farger er ikke lenger tilgjengelig på Linux konsollen med mindre en rammebuffer brukes. Hvis man ønsker å ha lyse farger uten en rammebuffer og kan leve uten karakterer som ikke tilhører språket hans, er det fortsatt mulig å bruke en språkspesifikk font med 256 glyffer, som illustrert nedenfor:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# End /etc/sysconfig/console
EOF
```

- Følgende eksempel illustrerer keymap sin autokonvertering fra ISO-8859-15 til UTF-8 og aktivering av døde nøkler i Unicode-modus:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
FONT="LatArCyrHeb-16 -m 8859-15"

# End /etc/sysconfig/console
EOF
```

- Noen tastaturmapper har døde taster (dvs. taster som ikke produserer en karakter av seg selv, men legger en aksent på karakteren som er produsert ved neste nøkkel) eller definer komposisjonsregler (som f.eks: «press Ctrl+. A E for å hente Æ» i standard tastaturopsett). Linux-6.13.4 tolker døde taster og komposisjonsregler i keymap riktig bare når kildetegnene som skal komponeres sammen ikke er multibyte. Denne mangelen påvirker ikke tastaturkart for europeiske språk, fordi aksenter legges til ASCII uten aksent tegn, eller to ASCII-tegn er satt sammen. Imidlertid, i UTF-8-modus er det et problem; f.eks. for det greske språket, hvor man noen ganger trenger å sette en aksent på bokstaven α . Løsningen er enten å unngå bruk av UTF-8, eller å installere X-vindussystemet som ikke har denne begrensningen i inndata håndtering.
- For kinesisk, japansk, koreansk og noen andre språk, Linux konsollen kan ikke konfigureres til å vise de nødvendige tegnene. Brukere som trenger slike språk bør installere et X Window System, fonter som dekker de nødvendige tegnområdene og den riktige inndatametoden (f.eks. SCIM, støtter et bredt utvalg av språk).



Notat

`/etc/sysconfig/console` filen kontrollerer kun lokaliseringen av Linux-tekstkonsollen. Det har ingenting med innstilling å gjøre riktig tastaturopsett og terminalfonter i X Window System, med ssh økter, eller med en seriell konsoll. I slike situasjoner, begrensninger nevnt i de to siste listepunktene ovenfor gjelder ikke.

9.6.6. Opprette filer ved oppstart

Noen ganger er det ønskelig å lage filer ved oppstart. For eksempel, `/tmp/.ICE-unix` mappen er ofte nødvendig. Dette kan gjøres ved å opprette en oppføring i `/etc/sysconfig/createfiles` konfigurasjonskriptet. Formatet til denne filen er innebygd i kommentarene til standard konfigurasjonsfil.

9.6.7. Konfigurering av sysklogd skriptet

`sysklogd` skriptet starter `syslogd` programmet som en del av System V initialisering. `-m 0` alternativet slår av det periodiske tidsstempelen `syslogd` skriver til loggfilene hvert 20. minutt som standard. Hvis du vil slå på dette periodiske tidsstempelmerket, rediger `/etc/sysconfig/rc.site` og definere variabelen `SYSKLOGD_PARMS` til ønsket verdi. For å fjerne alle parametere, sett variabelen til en nullverdi:

```
SYSKLOGD_PARMS=
```

Se `man syslogd` for flere alternativer.

9.6.8. rc.site filen

Den valgfrie `/etc/sysconfig/rc.site` filen inneholder innstillinger som angis automatisk for hvert SystemV oppstartsskript. Det kan alternativt angi verdiene spesifisert i `hostname`, `console`, og `clock` filer i `/etc/sysconfig/` mappen. Hvis tilknyttede variabler er til stede i begge disse separate filene og `rc.site`, verdiene i de skriptspesifikke filene har presedens.

`rc.site` inneholder også parametere som kan tilpasse andre aspekter av oppstartsprosessen. Stille inn `IPROMPT` variabelen vil aktivere selektiv kjøring av oppstartsskript. Andre alternativer er beskrevet i filkommentarene. Standardversjonen av filen er som følger:

```
# rc.site
# Optional parameters for boot scripts.

# Distro Information
# These values, if specified here, override the defaults
#DISTRO="Linux From Scratch" # The distro name
#DISTRO_CONTACT="lfs-dev@lists.linuxfromscratch.org" # Bug report address
#DISTRO_MINI="LFS" # Short name used in filenames for distro config

# Define custom colors used in messages printed to the screen

# Please consult `man console_codes` for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font. This does
# not affect framebuffer consoles

# These values, if specified here, override the defaults
#BRACKET="\033[1;34m" # Blue
#FAILURE="\033[1;31m" # Red
#INFO="\033[1;36m" # Cyan
#NORMAL="\033[0;39m" # Grey
#SUCCESS="\033[1;32m" # Green
#WARNING="\033[1;33m" # Yellow

# Use a colored prefix
# These values, if specified here, override the defaults
#BMPREFIX=" "
#SUCCESS_PREFIX="${SUCCESS} * ${NORMAL} "
#FAILURE_PREFIX="${FAILURE}*****${NORMAL} "
#WARNING_PREFIX="${WARNING} *** ${NORMAL} "

# Manually set the right edge of message output (characters)
# Useful when resetting console font during boot to override
# automatic screen width detection
#COLUMNS=120

# Interactive startup
#IPROMPT="yes" # Whether to display the interactive boot prompt
#itime="3" # The amount of time (in seconds) to display the prompt
```



```

# The total length of the distro welcome string, without escape codes
#wlen=$(echo "Welcome to ${DISTRO}" | wc -c )
#welcome_message="Welcome to ${INFO}${DISTRO}${NORMAL}"

# The total length of the interactive string, without escape codes
#ilen=$(echo "Press 'I' to enter interactive startup" | wc -c )
#i_message="Press '${FAILURE}I${NORMAL}' to enter interactive startup"

# Set scripts to skip the file system check on reboot
#FASTBOOT=yes

# Skip reading from the console
#HEADLESS=yes

# Write out fsck progress if yes
#VERBOSE_FSCK=no

# Speed up boot without waiting for settle in udev
#OMIT_UDEV_SETTLE=y

# Speed up boot without waiting for settle in udev_retry
#OMIT_UDEV_RETRY_SETTLE=yes

# Skip cleaning /tmp if yes
#SKIPTMPCLEAN=no

# For setclock
#UTC=1
#CLOCKPARAMS=

# For consolelog (Note that the default, 7=debug, is noisy)
#LOGLEVEL=7

# For network
#HOSTNAME=mylfs

# Delay between TERM and KILL signals at shutdown
#KILLDELAY=3

# Optional syslogd parameters
#SYSKLOGD_PARAMS="-m 0"

# Console parameters
#UNICODE=1
#KEYMAP="de-latin1"
#KEYMAP_CORRECTIONS="euro2"
#FONT="lat0-16 -m 8859-15"
#LEGACY_CHARSET=

```

9.6.8.1. Tilpasse oppstarts og avslutnings skriptene

LFS oppstartsskriptene starter opp og slår av et system på en rimelig effektiv måte, men det er noen få justeringer du kan gjøre i rc.site filen for å forbedre hastigheten enda mer og for å justere meldinger i henhold til dine preferanser. For å gjøre dette, juster innstillingene i `/etc/sysconfig/rc.site` filen ovenfor.

- Under oppstartsskriptet `udev`, er det et kall til **udev settle** som krever litt tid for å fullføres. Denne tiden kan være nødvendig eller ikke, avhengig av tilstedeværende enheter i systemet. Hvis du bare har enkle partisjoner og et enkelt ethernet kort, vil oppstartsprosessen sannsynligvis ikke trenge å vente på denne kommandoen. For å hopp over det, sett variabelen `OMIT_UDEV_SETTLE=y`.
- Oppstartsskriptet `udev_retry` kjører også **udev settle** som standard. Denne kommandoen er kun nødvendig som standard hvis `/var` mappen er separat montert. Dette er fordi klokken trenger filen `/var/lib/hwclock/adjtime`. Andre tilpasninger kan også må vente på at `udev` skal fullføres, men i mange installasjoner er det ikke behov for det. Hopp over kommandoen ved å sette variabelen `OMIT_UDEV_RETRY_SETTLE=y`.

- Som standard er filsystemkontrollene stille. Dette kan se ut til å være en forsinkelse under oppstartsprosessen. For å slå på **fsck** utdata, sett variabelen `VERBOSE_FSCK=y`.
- Ved omstart kan det være lurt å hoppe over filsystemkontrollen, **fsck**, helt. For å gjøre dette, opprett enten filen `/fastboot` eller start systemet på nytt med kommandoen `/sbin/shutdown -f -r now`. På den annen side kan du tvinge alle filsystemer til å bli kontrollert ved å opprette `/forcefsck` eller kjøre **shutdown** med `-F` parameter i stedet for `-f`.
Å sette variabelen `FASTBOOT=y` vil deaktivere **fsck** under oppstartsprosessen til den fjernes. Dette anbefales ikke på permanent basis.
- Normalt slettes alle filene i `/tmp` mappen ved oppstart. Avhengig av antall filer eller mapper som er tilstede, kan dette føre til en merkbar forsinkelse i oppstartsprosessen. For å hoppe over å fjerne disse filene, sett inn variabelen `SKIPTMPCLEAN=y`.
- Under avstengning, **init** programmet sender et **TERM** signal til hvert program det har startet (f.eks. `agetty`), venter på en tid (standard 3 sekunder), og sender hver prosess et **KILL** signal og venter en gang til. Denne prosessen gjentas i **sendsignals** skript for alle prosesser som ikke stenges av deres egne skript. Forsinkelse for **init** kan stilles inn ved å sende en parameter. For eksempel for å fjerne forsinkelsen i **init**, bruk `-t0` parameteren når du slår av eller starter på nytt (f.eks. `/sbin/shutdown -t0 -r now`). Forsinkelsen for **sendsignals** skriptet kan hoppes over ved å sette parameteren `KILLDELAY=0`.

9.7. Konfigurere systemlokaliteten

Noen miljøvariabler er nødvendige for morsmål støtte. Å sette dem riktig 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. Valgfrie 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 at `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.

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.

Skallprogrammet **/bin/bash** (her etter referert som «skallet») bruker en samling oppstartsfiler for å hjelpe med å lage miljøet å kjøre i. Hver fil har en bestemt bruk og kan påvirke pålogging og interaktive miljøer ulikt. Filene i `/etc` mappen gir globale innstillinger. Hvis tilsvarende filer finnes i hjemmemappen, kan de overstyre de globale innstillingene.

Et interaktivt påloggingsskall startes etter en vellykket pålogging, ved hjelp av **/bin/login**, ved å lese `/etc/passwd` filen. Et interaktivt ikke-påloggingsskall er startet på kommandolinjen (f.eks. `[prompt]$/bin/bash`). Et ikke-interaktivt skall er vanligvis tilstede når et skallskript kjører. Det er ikke-interaktivt fordi det behandler et skript og ikke venter for brukerinndata mellom kommandoer.

Opprett `/etc/profile` når de riktige lokale innstillingene har blitt bestemt for å angi ønsket lokalitet, men still inn `C.UTF-8` lokalitet i stedet hvis den kjøres i en Linux konsoll (for å hindre programmer fra å skrive ut tegn som Linux konsollen ikke kan gjengi):

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

for i in $(locale); do
    unset ${i%=*}
done

if [[ "$TERM" = linux ]]; then
    export LANG=C.UTF-8
else
    export LANG=<ll>_<CC>.<charmap><modifiers>
fi

# End /etc/profile
EOF
```

`c` (standard) og `en_US` (den anbefalte for engelske brukere i USA) er annerledes. `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 spørsmålsteget 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 `unknown 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` som 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 8-bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions
"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

9.9. 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
```

Kapittel 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
proc          /proc               proc       nosuid,noexec,nodev    0     0
sysfs         /sys                sysfs      nosuid,noexec,nodev    0     0
devpts        /dev/pts            devpts     gid=5,mode=620          0     0
tmpfs         /run                tmpfs      defaults                 0     0
devtmpfs      /dev                devtmpfs   mode=0755,nosuid        0     0
tmpfs         /dev/shm            tmpfs      nosuid,nodev             0     0
cgroup2       /sys/fs/cgroup      cgroup2    nosuid,noexec,nodev    0     0

# End /etc/fstab
EOF
```

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 [fstab\(5\)](#).

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 File systems -> Native Language Support 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 skiller 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 «Default NLS Option» (`CONFIG_NLS_DEFAULT`), «Default Remote NLS Option» (`CONFIG_SMB_NLS_DEFAULT`), «Default codepage for FAT» (`CONFIG_FAT_DEFAULT_CODEPAGE`), and «Default 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-6.13.4

Linux pakken inneholder Linux kjernen.

Omtrentlig byggetid: 0.4 - 32 SBU (typically about 2.5 SBU)

Nødvendig diskplass: 1.7 - 14 GB (typically about 2.3 GB)

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.



Viktig

Å bygge Linuxkjernen for første gang er en av de mest utfordrende oppgaver i LFS. Å få det riktig avhenger av den spesifikke maskinvaren for målsystemet og dine spesifikke behov. Det er nesten 12 000 konfigurasjonselementer som er tilgjengelige for kjernen, selv om bare en tredjedel av dem er nødvendig for de fleste datamaskiner. LFS redaktørene anbefaler at brukere som ikke er kjent med denne prosessen følger prosedyrene nedenfor ganske næye. Målet er å få et innledende system til et punkt hvor du kan logge inn på kommandolinjen når du starter på nytt senere i Seksjon 11.3, «Omstart av systemet» På dette punktet er optimering og tilpasning ikke et mål.

For generell informasjon om kjernekonfigurasjon se <https://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt>. Ytterligere informasjon om konfigurering og bygging av kjernen finner du på <https://andu.in.linuxfromscratch.org/LFS/kernel-nutshell/>. Disse referansene er litt utdatert, men gir likevel en rimelig oversikt over prosessen.

Hvis alt annet feiler, kan du be om hjelp på *lfs-support* mailingliste. Vær oppmerksom på at du må abonnere på listen for å unngå spam.

Forbered for kompilering ved å kjøre følgende kommando:

```
make mrproper
```

Dette sikrer at kjernetreet er helt rent. Kjerneteamet 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 `menuconfig` ncurses grensesnitt linjetegning på en UTF-8 linux tekstkonsoll.

Hvis brukt, sørg for å erstatte `<host_LANG_value>` med verdien av `$LANG` variabel fra verten din. Du kan alternativt bruke vertens verdi av `$LC_ALL` eller `$LC_CTYPE`.

make menuconfig

Dette starter et ncurses menydrevet grensesnitt. For andre (grafiske) grensesnitt, skriv **make help**.



Notat

Et godt utgangspunkt for å sette opp kjernekonfigurasjonen er å kjøre **make defconfig**. Dette vil sette basekonfigurasjonen 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 --->
[ ] Compile the kernel with warnings as errors                [WERROR]
CPU/Task time and stats accounting --->
[*] Pressure stall information tracking                        [PSI]
[ ]   Require boot parameter to enable pressure stall information tracking
... [PSI_DEFAULT_DISABLED]
< > Enable kernel headers through /sys/kernel/kheaders.tar.xz [IKHEADERS]
[*] Control Group support --->                             [CGROUPS]
[*] Memory controller                                       [MEMCG]
[ ] Configure standard kernel features (expert users) --->  [EXPERT]

Processor type and features --->
[*] Build a relocatable kernel                              [RELOCATABLE]
[*]   Randomize the address of the kernel image (KASLR)      [RANDOMIZE_BASE]

General architecture-dependent options --->
[*] Stack Protector buffer overflow detection                [STACKPROTECTOR]
[*]   Strong Stack Protector                                [STACKPROTECTOR_STRONG]

Device Drivers --->
Generic Driver Options --->
[ ] Support for uevent helper                                [UEVENT_HELPER]
[*] Maintain a devtmpfs filesystem to mount at /dev          [DEVTMPFS]
[*]   Automount devtmpfs at /dev, after the kernel mounted the rootfs
... [DEVTMPFS_MOUNT]

Firmware Drivers --->
[*] Mark VGA/VBE/EFI FB as generic system framebuffer       [SYSFB_SIMPLEFB]

Graphics support --->
<*> Direct Rendering Manager (XFree86 4.1.0 and higher DRI support) --->
... [DRM]
[*]   Display a user-friendly message when a kernel panic occurs
... [DRM_PANIC]
(kmsg) Panic screen formatter                              [DRM_PANIC_SCREEN]
Supported DRM clients --->
[*] Enable legacy fbdev support for your modesetting driver
... [DRM_FBDEV_EMULATION]
<*>   Simple framebuffer driver                            [DRM_SIMPLEDRM]
Console display driver support --->
[*] Framebuffer Console support                            [FRAMEBUFFER_CONSOLE]

```

Aktiver noen tilleggsfunksjoner hvis du bygger et 64-bit system. Hvis du bruker menuconfig, aktiver dem i rekkefølgen `CONFIG_PCI_MSI` først, deretter `CONFIG_IRQ_REMAP`, til sist `CONFIG_X86_X2APIC` fordi et alternativ kun vises etter at avhengighetene er valgt.

```

Processor type and features --->
[*] Support x2apic                                          [X86_X2APIC]

Device Drivers --->
[*] PCI support --->                                       [PCI]
[*] Message Signaled Interrupts (MSI and MSI-X)           [PCI_MSI]
[*] IOMMU Hardware Support --->                           [IOMMU_SUPPORT]
[*] Support for Interrupt Remapping                        [IRQ_REMAP]

```

Hvis du bygger et 32-bits system som kjører på en maskinvare med mer enn 4 GB RAM, juster konfigurasjonen slik at kjernen vil kunne bruke opptil 64 GB fysisk RAM:

```

Processor type and features --->
High Memory Support --->
(X) 64GB                                                    [HIGHMEM64G]

```

Hvis partisjonen for LFS-systemet er i en NVME SSD (dvs enhetsnoden for partisjonen er `/dev/nvme*` i stedet for `/dev/sd*`), aktivere NVME støtte ellers vil ikke LFS systemet starte:

```
Device Drivers --->
  NVME Support --->
    <*> NVM Express block device [BLK_DEV_NVME]
```

Det er flere andre alternativer som kan være ønsket avhengig av kravene til systemet. For en liste over nødvendige alternativer for BLFS pakker, se *BLFS Indeks over kjerneinnstillinger*.



Notat

Hvis vertsmaskinvaren din bruker UEFI og du ønsker å starte opp LFS systemet med det, bør du justere noen kjernekonfigurasjon som på følgende *BLFS side selv om du bruker UEFI oppstartslaster fra vertsdistroen*.

Begrunnelsen for de ovennevnte konfigurasjonselementene:

Randomize the address of the kernel image (KASLR)

Aktiver ASLR for kjernebilde for å redusere noen angrep basert på faste adresser til sensitive data eller kode i kjernen.

Compile the kernel with warnings as errors

Dette kan forårsake bygningsfeil hvis kompilatoren og/eller konfigurasjonen er forskjellig fra kjernens utviklere.

Enable kernel headers through /sys/kernel/kheaders.tar.xz

Dette vil kreve **cpio** for å bygge kjernen. **cpio** er ikke installert av LFS.

Configure standard kernel features (expert users)

Dette vil gjøre at noen alternativer vises i konfigurasjonsgrensesnittet, men å endre disse alternativene kan være farlig. Ikke bruk dette med mindre du vet hva du gjør.

Strong Stack Protector

Aktiver SSP for kjernen. Vi har aktivert det for hele brukerplassen med `--enable-default-ssp` konfigureringen for GCC, men kjernen bruker ikke GCC standardinnstillingen for SSP. Vi aktiverer det eksplisitt her.

Support for uevent helper

Å ha dette alternativet satt kan forstyrre enhetens behandling ved bruk av Udev.

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.

Automount devtmpfs at /dev

Dette vil montere kjernevisningen til enhetene på `/dev` ved bytte til rotfilssystem rett før start av init.

Vis en brukervennlig melding når en kjernepanikk oppstår

Dette vil få kjernen til å vise meldingen riktig i tilfelle kjernepanikk oppstår og en kjørende DRM driver støtter å gjøre det. Uten dette ville det blitt mer vanskelig å diagnostisere panikk: hvis ingen DRM driver kjører, vi ville være på VGA konsollen som bare kan inneholde 24 linjer og den relevante kjerne meldingen skylles ofte bort; hvis en DRM driveren kjører, er skjermen ofte fullstendig rotete i panikk. Fra og med Linux-6.12, ingen av de dedikerte driverne for mainstream GPU modeller støtter virkelig dette, men det støttes av «Simple framebuffer driver» som kjører på VESA (eller EFI) rammebuffer før den dedikerte GPU driveren er lastet. Hvis den dedikerte GPU driveren er bygget som en modul (i stedet for en del av kjernebildet) og ingen initramfs er brukt, vil denne funksjonaliteten fungere helt fint før roten av filsystemet er montert og det er allerede nok

for å gi informasjon om de fleste LFS konfigurasjonsfeil som forårsaker en panikk (for eksempel en feil `root=` innstilling i Seksjon 10.4, «Bruke GRUB til å sette opp oppstartsprosessen»).

Panikkskjermformater

Still inn denne `kmsg` for å sikre at de siste kjernemeldingslinjer vises når en kjernepanikk oppstår. Standard, bruker, ville få kjernen til å vise bare en «brukervennlig» panikkmelding som ikke er nyttig å diagnostisere. Det tredje valget, `qr_code`, ville få kjernen til å komprimere de siste kjernemeldingslinjene i en QR kode og vise den. QR koden kan inneholde flere meldingslinjer enn ren tekst og den kan dekodes med en ekstern enhet (som en smarttelefon). Men det krever en Rust-kompilator som LFS ikke leverer.

Merk VGA/VBE/EFI FB som generisk systemrammebuffer Og Enkel framebuffer driver

Disse gjør det mulig å bruke VESA rammebufferen (eller EFI framebuffer hvis du starter opp LFS systemet via UEFI) som en DRM enhet. VESA rammebufferen vil bli satt opp av GRUB (eller EFI framebuffer vil bli satt opp av UEFI fastvaren), så DRM panikken handleren kan fungere før den GPU spesifikke DRM driveren er lastet.

Aktiver eldre fbdev støtte for modusinnstillingsdriveren Og Framebuffer Console support

Disse er nødvendige for å vise Linux konsollen på en GPU drevet av en DRI driver (Direct Rendering Infrastructure). Siden `CONFIG_DRM` (Direct Rendering Manager) er aktivert, bør vi aktivere disse to alternativene også, ellers får vi se en tom skjerm når DRI driveren er lastet inn.

Support x2apic

Støtte for å kjøre avbruddskontrolleren for 64-bit x86 prosessorer i x2APIC-modus. x2APIC kan være aktivert av fastvare på 64-bit x86-systemer, og en kjerne uten dette alternativet aktivert vil få panikk ved oppstart hvis x2APIC er aktivert av fastvare. Dette alternativet har ingen effekt, men gjør heller ingen skade hvis x2APIC er deaktivert av fastvare.

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-6.13.4` 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 Seksjon 9.3, «Oversikt over enhets- og modulhåndtering» og kjerne dokumentasjon i `linux-6.13.4/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.



Obs

Hvis du har bestemt deg for å bruke et separat `/boot` partisjon for LFS systemet (kanskje dele en `/boot` partisjon med vertsdistroen), filene som er kopiert nedenfor, skal gå dit. Den enkleste måten å gjøre det er å opprette oppfæringen for `/boot` i `/etc/fstab` først (les forrige seksjon for detaljer), utfør deretter følgende kommando som `root` bruker i *chroot environment*:

```
mount /boot
```

Stien til enhetsnoden er utelatt i kommandoen fordi **mount** kan lese den fra `/etc/fstab`.

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-6.13.4-lfs-12.3
```

`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-6.13.4
```

Kjernens konfigurasjonsfil `.config` produsert av **make menuconfig** steget ovenfor inneholder alle konfigurasjonsvalgene for kjernen som nettopp ble kompilert. Det er en god ide å beholde denne filen for fremtidig referanse:

```
cp -iv .config /boot/config-6.13.4
```

Installer dokumentasjonen for Linux kjernen:

```
cp -r Documentation -T /usr/share/doc/linux-6.13.4
```

Det er viktig å merke seg at filene i kjernebildens 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 kildetrete ofte beholdt i lang tid. På grunn av dette er det en sjanse at hvilken bruker-ID pakken brukte vil bli tildelt noen på maskinen. Den personen ville da ha skrivetilgang til kjernens kilde.



Notat

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 kjernebildet etter at den nybygde kjernen er installert.

Hvis kjernebildet skal beholdes, kjør **chown -R 0:0** på `linux-6.13.4` mappen å sikre at alle filer eies av brukeren *root*.

Hvis du oppdaterer konfigurasjonen og gjenoppbygger kjernen fra et beholdt kjernebilde, bør du normalt **ikke** kjøre **make mrproper** kommandoen. Kommandoen ville rense `.config` filen og alle `.o` filer fra forrige bygg. Til tross for at det er enkelt å gjenopprette `.config` fra kopien i `/boot`, å rense alle `.o` filer er fortsatt bortkastet: for en enkel konfigurasjonsendring, må ofte bare noen få `.o` filer (om)bygges og kjernebyggesystemet vil riktig hoppe over andre `.o` filer hvis de ikke er renset.

På den annen side, hvis du har oppgradert GCC, bør du kjøre **make clean** for å rense alle `.o` filer fra forrige bygg, ellers kan det nye bygget mislykkes.



Advarsel

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.

10.3.2. Konfigurere rekkefølgen på Linux modullasting

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-6.13.4`, `vmlinuz-6.13.4-lfs-12.3`, og `System.map-6.13.4`
Installerte mapper: `/lib/modules`, `/usr/share/doc/linux-6.13.4`

Korte beskrivelser

<code>config-6.13.4</code>	Inneholder alle konfigurasjonsvalgene for kjernen
<code>vmlinuz-6.13.4-lfs-12.3</code>	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
<code>System.map-6.13.4</code>	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



Notat

Hvis systemet ditt har UEFI støtte og du ønsker å starte LFS med UEFI, bør du hoppe over instruksjonene på denne siden, men likevel lære deg syntaksen til `grub.cfg` og metoden for å spesifisere en partisjon i filen fra denne siden, og konfigurere GRUB med UEFI støtte ved å bruke instruksjonene gitt i *BLFS siden*.

10.4.1. Introduksjon



Advarsel

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 partisjonsnummeret. Harddisknummeret starter fra null, men partisjonsnummeret starter fra en 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 harddisk. 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 kan hvert bygg, enten LFS eller en kommersiell distro, 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`.

Å la `/boot` være på nåværende LFS partisjon vil også virke, men konfigurasjonen for flere systemer er mer vanskelig.

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:



Advarsel

Følgende kommando vil overskrive gjeldende oppstartslaster. Ikke kjør kommandoen hvis dette ikke er ønsket, for eksempel hvis du bruker en tredjeparts oppstartsbehandler for å administrere Master Boot Record (MBR).

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



Notat

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 Kapittel 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 part_gpt
insmod ext2
set root=(hd0,2)
set gfxpayload=1024x768x32

menuentry "GNU/Linux, Linux 6.13.4-lfs-12.3" {
    linux /boot/vmlinuz-6.13.4-lfs-12.3 root=/dev/sda2 ro
}
EOF
```

insmod kommandoer laster inn GRUB moduler navngitt `part_gpt` og `ext2`. Til tross for navngivningen, støtter `ext2` faktisk `ext2`, `ext3`, og `ext4` filsystemer. **grub-install** kommandoen har bygd inn noen moduler inn i hoveddelen av GRUB bildet (installert i MBR- eller GRUB BIOS-partisjonen) for å få tilgang til de andre modulene (i `/boot/grub/i386-pc`) uten et kylling-eller-egg-problem, så med en typisk konfigurasjon er disse to modulene allerede innebygd og de to **insmod** kommandoer vil ikke gjøre noe. Men de gjør ingen skade uansett, og det kan være nødvendig med noen sjeldne konfigurasjoner.

set gfxpayload=1024x768x32 kommandoen setter oppløsning og fargedybde på VESA rammebufferen som skal sendes til kjernen. Det er nødvendig for kjernens SimpleDRM driver å bruke VESA rammebuffer. Du kan bruke en annen verdi for oppløsning eller fargedybde som passer best for skjermen din.



Notat

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.



Notat

GRUB betegnelsen for en partisjon kan endres hvis du la til eller fjernet noen disk (inkludert flyttbare disk som USB-enheter). Endringen kan forårsake oppstartsfeil pga `grub.cfg` refererer til noen «gamle» betegnelser. Hvis du ønsker å unngå et slikt problem, kan du bruke UUID for partisjon og filsystem i stedet for GRUB betegnelser til å angi en partisjon. Kjør **`lsblk -o UUID,PARTUUID,PATH,MOUNTPOINT`** for å vise UUID-ene til filsystemene dine (i `UUID` kolonnen) og partisjoner (i `PARTUUID` kolonnen). Bytt deretter ut `set root=(hdx,y)` med `search --set=root --fs-uuid <UUID for filsystemet der kjernen er installert>`, og erstatt `root=/dev/sda2` med `root=PARTUUID=<UUID for partisjonen der LFS er bygget>`.

Merk at partisjonens UUID og filsystemets UUID i denne partisjonen er helt annerledes. Noen nettressurser kan instruere deg om å bruke `root=UUID=<filesystem UUID>` i stedet for `root=PARTUUID=<partition UUID>`, men å gjøre det vil kreve en `initramfs` som er utenfor omfanget av LFS.

Navnet på enhetsnoder for en partisjon i `/dev` kan også endres (mer usannsynlig enn GRUB betegnelsesendring). Du kan også bytte ut stier til enhetsnoder som `/dev/sda1` med `PARTUUID=<partition UUID>`, i `/etc/fstab`, for å unngå en potensiell oppstartsfeil i tilfelle enhetens nodenavn er endret.

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.



Obs

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.

Kapittel 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 opprette 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 12.3 > /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 nye 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="12.3"
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="12.3"
ID=lfs
PRETTY_NAME="Linux From Scratch 12.3"
VERSION_CODENAME="<your name here>"
HOME_URL="https://www.linuxfromscratch.org/lfs/"
RELEASE_TYPE="stable"
EOF
```

Pass på å tilpasse feltene 'DISTRIB_CODENAME' og 'VERSION_CODENAME' for å gjøre systemet unikt ditt.

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. Det er imidlertid fortsatt et par ting å sjekke. Her er noen forslag:

- Installer evt *firmware* som er nødvendig hvis kjernedriver for maskinvaren din krever noen fastvarefiler for å fungere skikkelig.
- Sørg for at et passord er angitt for `root` brukeren.
- 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
- /etc/sysconfig/ifconfig.eth0

Nå som vi har sagt det, la oss gå videre til å starte opp vår skinnende nye LFS installasjon for første gang! *Først gå ut av chroot-miljøet:*

```
logout
```

Deretter avmontere de virtuelle filsystemene:

```
umount -v $LFS/dev/pts
mountpoint -q $LFS/dev/shm && umount -v $LFS/dev/shm
umount -v $LFS/dev
umount -v $LFS/run
umount -v $LFS/proc
umount -v $LFS/sys
```

Hvis flere partisjoner ble opprettet, avmonter den andre partisjoner før du demonterer den viktigste, slik som dette:

```
umount -v $LFS/home
umount -v $LFS
```

Avmonter selve LFS filsystemet:

```
umount -v $LFS
```

Nå start nå systemet på nytt.

Forutsatt at GRUB oppstartslasteren ble satt opp som skissert tidligere, er menyen satt til å starte opp *LFS 12.3* automatisk.

Når omstarten er fullført, er LFS systemet klart til bruk. Hva du vil se er en enkel «login: » ledetekst. På dette tidspunktet kan du fortsett til *BLFS boken* hvor du kan legge til mer programvare som passer dine behov.

Hvis omstarten **ikke** er vellykket, er det på tide å feilsøke. For tips om hvordan du løser innledende oppstartsproblemer, se <https://www.linuxfromscratch.org/lfs/troubleshooting.html>.

11.4. Tilleggsressurser

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:

- *LFS sikkerhetsråd*

Dette er en liste over sikkerhetssårbarheter oppdaget i LFS-bok etter at den er utgitt.

- *E-postliste for sikkerhet med åpen kildekode*

Dette er en e-postliste for diskusjon av sikkerhetsfeil, konsepter og praksiser i åpen kildekodefellesskapet.

- 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 Kapittel 1 - E-postlister 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 mansider. Den ligger på <https://tldp.org/>.

11.5. Komme i gang etter LFS

11.5.1. Bestemme hva du skal gjøre videre

Nå som LFS er fullført og du har et oppstartbart system, hva gjør du? Det neste trinnet er å bestemme hvordan den skal brukes. Generelt er det to brede kategorier å vurdere: arbeidsstasjon eller server. Faktisk disse kategoriene utelukker ikke hverandre. Applikasjonene som trengs for hver kategori kan kombineres til et enkelt system, men la oss se på dem separat for nå.

En server er den enklere kategorien. Vanligvis består dette av en nettserver som *Apache HTTP Server* og en databaseserver som f.eks *MariaDB*. Men andre tjenester er mulig. Operativsystemet innebygd i en engangsenhet faller inn i denne kategorien.

På den annen side er en arbeidsstasjon mye mer kompleks. Generelt krever den et grafisk brukermiljø som f.eks *LXDE*, *XFCE*, *KDE*, eller *Gnome* basert på et grunnleggende *grafisk miljø* og flere grafisk baserte applikasjoner som f.eks *Firefox nettleser*, *Thunderbird e-postklient*, eller *LibreOffice kontorpakke*. Disse applikasjonene krever mange (flere hundre avhengig av ønskede funksjoner) flere pakker med støtteapplikasjoner og biblioteker.

I tillegg til ovennevnte er det et sett med applikasjoner for systemstyring for alle typer systemer. Disse applikasjonene er alle i BLFS boken. Ikke alle pakker er nødvendige i hvert miljø. F.eks *dhcpcd*, er normalt ikke egnet for en server og *trådløse verktøy*, er normalt bare nyttige for et bærbart system.

11.5.2. Arbeide i et grunnleggende LFS miljø

Når du starter opp i LFS, har du alle interne verktøy for å bygge tilleggspakker. Dessverre er brukermiljøet ganske sparsomt. Det er et par måter å forbedre dette på:

11.5.2.1. Arbeide fra LFS verten i chroot

Denne metoden gir et komplett grafisk miljø hvor fulle funksjoner for nettleser og kopier/lim inn er tilgjengelige. Denne metoden lar deg laste ned applikasjoner som vertens versjon av *wget* for å laste ned pakkekilder til et sted som er tilgjengelig når du arbeider i chroot miljøet.

For å kunne bygge pakker på riktig måte i chroot, må du også huske å montere de virtuelle filsystemene hvis de ikke allerede er montert. En måte å gjøre dette på er å lage et skript på **VERTS** systemet:

```
cat > ~/mount-virt.sh << "EOF"
#!/bin/bash

function mountbind
{
    if ! mountpoint $LFS/$1 >/dev/null; then
        $SUDO mount --bind /$1 $LFS/$1
        echo $LFS/$1 mounted
    else
        echo $LFS/$1 already mounted
    fi
}

function mounttype
{
    if ! mountpoint $LFS/$1 >/dev/null; then
        $SUDO mount -t $2 $3 $4 $5 $LFS/$1
        echo $LFS/$1 mounted
    else
        echo $LFS/$1 already mounted
    fi
}

if [ $EUID -ne 0 ]; then
    SUDO=sudo
else
    SUDO=""
fi

if [ x$LFS == x ]; then
    echo "LFS not set"
    exit 1
fi

mountbind dev
mounttype dev/pts devpts devpts -o gid=5,mode=620
mounttype proc    proc    proc
mounttype sys     sysfs   sysfs
mounttype run     tmpfs   run
if [ -h $LFS/dev/shm ]; then
    install -v -d -m 1777 $LFS$(realpath /dev/shm)
else
    mounttype dev/shm tmpfs tmpfs -o nosuid,nodev
fi

#mountbind usr/src
#mountbind boot
#mountbind home
EOF
```

Merk at de tre siste kommandoene i skriptet er kommentert ut. Disse er nyttige hvis disse mappene er montert som separate partisjoner på vertssystemet og vil bli montert ved oppstart av det fullførte LFS/BLFS-systemet.

Skriptet kan kjøres med **bash ~/mount-virt.sh** som enten en vanlig bruker (anbefalt) eller som **root**. Hvis du kjører som vanlig bruker, er **sudo** nødvendig på vertssystemet.

Et annet problem påpekt av skriptet er hvor du skal lagre nedlastede pakkefiler. Denne plasseringen er vilkårlig. Det kan være i en vanlig brukers hjemmekatalog som **~/sources** eller på en global plassering som **/usr/src**. Vær anbefaling er å ikke blande BLFS kilder og LFS kilder i (fra chroot-miljøet) **/sources**. I alle fall, pakkene må være tilgjengelig inne i chroot-miljøet.

En siste bekvemmelighet som presenteres her er å strømlinjeforme prosessen for å gå inn i chroot-miljøet. Dette kan gjøres med et alias plassert i en brukers `~/.bashrc`-fil på vertssystemet:

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

Dette aliaset er litt vanskelig på grunn av sitering og nivåer av omvendt skråstrek. Det må være på en linje. Kommandoen ovenfor er delt i to for presentasjonsformål.

11.5.2.2. Arbeide eksternt via ssh

Denne metoden gir også et fullstendig grafisk miljø, men først kreves installasjon av *sshd* på LFS systemet, vanligvis i chroot. Det krever også en datamaskin nummer to. Denne metoden har fordelen av å være enkel ved å ikke kreve kompleksiteten til chroot-miljøet. Den bruker også din LFS bygde kjerne for alle tilleggspakker og gir fortsatt et komplett system for å installere pakker.

Du kan bruke **scp** kommando for å laste opp pakke kilder som skal bygges inn i LFS systemet. Hvis du vil laste ned kildene direkte på LFS systemet, installer *libtasn1*, *p11-kit*, *make-ca*, og *wget* i chroot (eller last opp kildene deres ved å bruke **scp** etter oppstart av LFS systemet).

11.5.2.3. Arbeide fra LFS kommandolinjen

Denne metoden krever installasjon av *libtasn1*, *p11-kit*, *make-ca*, *wget*, *gpm*, og *links* (eller *lynx*) i chroot og deretter omstart i det nye LFS systemet. På dette punktet har standardsystemet seks virtuelle konsoller. Veksling mellom konsoller er like enkelt som å bruke **Alt+Fx** tastekombinasjoner hvor **Fx** er mellom **F1** og **F6**. Kombinasjonene **Alt+←** og **Alt+→** vil også endre konsollen.

På dette tidspunktet kan du logge på to forskjellige virtuelle konsoller og kjøre *links* eller *lynx* nettleseren i den ene konsollen og *bash* i den andre. GPM tillater da å kopiere kommandoer fra nettleseren med venstre museknapp, bytte konsoller, og lime inn i den andre konsollen.



Notat

Som en sidenotat kan bytting av virtuelle konsoller også gjøres fra en X Window forekomst med **Ctrl+Alt+Fx** tastekombinasjon, men kopieringsoperasjonen med mus fungerer ikke mellom det grafiske grensesnittet og en virtuell konsoll. Du kan gå tilbake til X Window-skjermen med **Ctrl+Alt+Fx** kombinasjonen, hvor **Fx** vanligvis er **F1** men kan være **F7**.

Del V. Vedlegg

Tillegg 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 nettpat (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 eksternt 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	Trådløkal 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 Identifier)
VC	Virtuell konsoll (Virtual Console)
VGA	Videografikkmatrise (Video Graphics Array)
VT	Virtuell terminal (Virtual Terminal)

Tillegg 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 opp til 2008
- *Jean-Philippe Mengual* <jmengual@linuxfromscratch.org> – Fransk LFS oversettelsesprosjekt 2008-2016
- *Julien Lepiller* <jlepiller@linuxfromscratch.org> – Fransk LFS oversettelsesprosjekt 2017-nåtid
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Portugisisk LFS oversettelsesprosjekt
- *Jamenson Espindula* <jafesp@gmail.com> – Portugisisk LFS oversettelsesprosjekt 2022-nåværende
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – Tysk 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 Phermawang* <satit@wbac.ac.th> – lfs.phayoune.org mirror
- *Shizunet Co.,Ltd.* <info@shizu-net.jp> – lfs.mirror.shizu-net.jp 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, Hints and Patches Project Maintainer
- *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> – Nettsteds 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-skripts vedlikeholder
- *Randy McMurphy* <randy@linuxfromscratch.org> – BLFS Prosjektleder, LFS-redaktør
- *Dan Nicholson* <dnicholson@linuxfromscratch.org> – LFS og BLFS Redaktør
- *Alexander E. Patrakov* <alexander@linuxfromscratch.org> – LFS Technical Writer, 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 Prosjekt Medleder
- *Greg Schafer* <gschafer@zip.com.au> – LFS teknisk skribent og Arkitekt for neste generasjons 64-biters byggemetode
- *Jesse Tie-Ten-Quee* – LFS Teknisk forfatter
- *James Robertson* <jwrober@linuxfromscratch.org> – Bugzilla vedlikeholder
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – BLFS bok Redaktør, Hints and Patches Prosjekt Leder
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – LFS teknisk Forfatter, Bugzilla vedlikeholder, LFS-Bootscripts vedlikeholder
- *Zack Winkles* <zwinkles@gmail.com> – LFS Teknisk forfatter

Tillegg 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 fjerde. 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, som 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.

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, Libcap, og Patch
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 og Coreutils
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: Coreutils
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:	<i>Xorg</i>

Bc

Installasjonen avhenger av:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, og Readline
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, Pkgconf, Sed, Texinfo, Zlib, og Zstd
Påkrevd ved kjøretid:	Glibc, Zlib, og Zstd
Testpakke avhenger av:	DejaGNU og Expect
Må installeres før:	Ingen
Valgfrie avhengigheter:	<i>Elfutils</i> og <i>Jansson</i>

Bison

Installasjonen avhenger av:	Bash, Binutils, Coreutils, Diffutils, 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:	<i>Doxygen</i>

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 og Libel
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:	<i>libsubunit</i> og <i>patchutils</i>

Coreutils

Installasjonen avhenger av:	Autoconf, Automake, 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, Findutils, Man-DB, og Udev
Valgfrie avhengigheter:	<i>Expect.pm</i> og <i>IO::Tty</i>

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, Pkgconf, 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

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:	<i>Tk</i>

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:	<i>libseccomp</i>

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

Flit-Core

Installasjonen avhenger av:	Python
Påkrevd ved kjøretid:	Python
Testpakke avhenger av:	Ingen testpakke tilgjengelig
Må installeres før:	Wheel
Valgfrie avhengigheter:	<i>pytest</i> og <i>testpath</i>

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:	<i>libsigsegv</i>

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:	<i>GDC</i> , <i>GNAT</i> , og <i>ISL</i>

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:	<i>libunistring</i> and <i>libxml2</i>

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:	<i>PCRE2</i> og <i>libsigsigv</i>

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
Valgfrie avhengigheter:	<i>ghostscript</i> og <i>Uchardet</i>

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, and 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

lana-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, Pkgconf, og Zlib
Påkrevd ved kjøretid:	Bash, Coreutils, Glibc, Libcap, Libelf, og Zlib
Testpakke avhenger av:	Ingen testpakke tilgjengelig
Må installeres før:	Ingen
Valgfrie avhengigheter:	<i>Berkeley DB, iptables, libbpf, libmnl, og libtirpc</i>

Jinja2

Påkrevd ved kjøretid:	MarkupSafe og Python
Testpakke avhenger av:	Ingen testpakke tilgjengelig
Må installeres før:	Udev
Valgfrie avhengigheter:	Ingen

Kbd

Installasjonen avhenger av:	Bash, Binutils, Bison, 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:	<i>Linux-PAM</i>

Kmod

Installasjonen avhenger av:	Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, OpenSSL, Pkgconf, Sed, Xz, og Zlib
Påkrevd ved kjøretid:	Glibc, Xz, og Zlib
Testpakke avhenger av:	Ingen testpakke tilgjengelig
Må installeres før:	Udev
Valgfrie avhengigheter:	<i>scdoc</i> (for manualsider)

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:	<i>PCRE2</i> eller <i>PCRE</i>

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:	<i>Linux-PAM</i>

Libelf

Installasjonen avhenger av:	Bash, Binutils, Bzip2, Coreutils, GCC, Glibc, Make, Xz, Zlib, og Zstd
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 og Pkgconf
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

Libxcrypt

Installasjonen avhenger av:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Perl, og Sed
Påkrevd ved kjøretid:	Glibc
Testpakke avhenger av:	Ingen
Må installeres før:	Perl, Python, Shadow, og Udev
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:	<i>cpio</i> , <i>LLVM</i> (med Clang), og <i>Rust-bindgen</i>

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

Lz4

Installasjonen avhenger av:	Bash, Binutils, Coreutils, GCC, Glibc, and Make
Påkrevd ved kjøretid:	Glibc
Testpakke avhenger av:	Python
Må installeres før:	Zstd
Valgfrie avhengigheter:	None

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:	<i>libsigsegv</i>

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:	<i>Guile</i>

Man-DB

Installasjonen avhenger av:	Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Pkgconf, 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:	<i>libseccomp</i> og <i>po4a</i>

Man-Pages

Installasjonen avhenger av:	Bash, Coreutils, Make, og Sed
Påkrevd ved kjøretid:	Ingen
Testpakke avhenger av:	Ingen testpakke tilgjengelig
Må installeres før:	Ingen
Valgfrie avhengigheter:	Ingen

MarkupSafe

Installasjonen avhenger av:	Python, Setuptools, og Wheel
Påkrevd ved kjøretid:	Python
Testpakke avhenger av:	Ingen testpakke tilgjengelig
Må installeres før:	Jinja2
Valgfrie avhengigheter:	Ingen

Meson

Installasjonen avhenger av:	Ninja, Python, Setuptools, og Wheel
Påkrevd ved kjøretid:	Python
Testpakke avhenger av:	Ingen testpakke tilgjengelig
Må installeres før:	Udev
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:	<i>cmake</i>
Må installeres før:	Meson
Valgfrie avhengigheter:	<i>Asciidoc</i> , <i>Doxygen</i> , <i>Emacs</i> , og <i>re2c</i>

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, Linux, og Udev
Valgfrie avhengigheter:	Ingen

Patch

Installasjonen avhenger av:	Attr, Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, og Sed
Påkrevd ved kjøretid:	Attr og Glibc
Testpakke avhenger av:	Diffutils
Må installeres før:	Ingen
Valgfrie avhengigheter:	<i>Ed</i>

Perl

Installasjonen avhenger av:	Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Libxcrypt, Make, Sed, og Zlib
Påkrevd ved kjøretid:	GDBM, Glibc, og Libxcrypt
Testpakke avhenger av:	Iana-Etc, Less, og Procps-ng
Må installeres før:	Autoconf
Valgfrie avhengigheter:	<i>Berkeley DB</i>

Pkgconf

Installasjonen avhenger av:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, og Sed
Påkrevd ved kjøretid:	Glibc
Testpakke avhenger av:	Ingen
Må installeres før:	Binutils, E2fsprogs, IProute2, Kmod, Man-DB, Procps-ng, Python, Udev, and Util-linux
Valgfrie avhengigheter:	Ingen

Procps-ng

Installasjonen avhenger av:	Bash, Binutils, Coreutils, GCC, Glibc, Make, Ncurses, og Pkgconf
Påkrevd ved kjøretid:	Glibc
Testpakke avhenger av:	DejaGNU
Må installeres før:	Ingen
Valgfrie avhengigheter:	<i>elogind</i>

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, Libxcrypt, Make, Ncurses, OpenSSL, Pkgconf, Sed, og Util-linux
Påkrevd ved kjøretid:	Bzip2, Expat, Gdbm, Glibc, Libffi, Libxcrypt, Ncurses, OpenSSL, og Zlib
Testpakke avhenger av:	GDB og Valgrind
Må installeres før:	Ninja
Valgfrie avhengigheter:	<i>Berkeley DB, libnsl, SQLite, og Tk</i>

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, Bc, 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

Setuptools

Installasjonen avhenger av:	Python og Wheel
Påkrevd ved kjøretid:	Python
Testpakke avhenger av:	Ingen testpakke tilgjengelig
Må installeres før:	Jinja2, MarkupSafe, og Meson
Valgfrie avhengigheter:	Ingen

Shadow

Installasjonen avhenger av:	Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Libcap, Libxcrypt, Make, og Sed
Påkrevd ved kjøretid:	Glibc og Libxcrypt
Testpakke avhenger av:	Ingen testpakke tilgjengelig
Må installeres før:	Coreutils
Valgfrie avhengigheter:	<i>CrackLib og Linux-PAM</i>

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

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

Udev

Installasjonen avhenger av:	Acl, Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Gperf, Grep, Jinja2, Libcap, Libxcrypt, Meson, OpenSSL, Pkgconf, Sed, Util-linux, og Zstd
Påkrevd ved kjøretid:	Acl, Glibc, Libcap, OpenSSL, og Util-linux
Testpakke avhenger av:	Ingen
Må installeres før:	Util-linux
Valgfrie avhengigheter:	Ingen

Util-linux

Installasjonen avhenger av:	Bash, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Pkgconf, Sed, Udev, og Zlib
Påkrevd ved kjøretid:	Glibc, Ncurses, Readline, Udev, og Zlib
Testpakke avhenger av:	Ingen
Må installeres før:	Ingen
Valgfrie avhengigheter:	<i>Asciidoctor</i> , <i>Libcap-NG</i> , <i>libeconf</i> , <i>libuser</i> , <i>libutempter</i> , <i>Linux-PAM</i> , <i>smartmontools</i> , <i>po4a</i> , og <i>slang</i>

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:	<i>Xorg, GTK+2, LessTif, Ruby, og GPM</i>

Wheel

Installasjonen avhenger av:	Python og Flit-core
Påkrevd ved kjøretid:	Python
Testpakke avhenger av:	Ingen testpakke tilgjengelig
Må installeres før:	Jinja2, MarkupSafe, Meson, og Setuptools
Valgfrie avhengigheter:	Ingen

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:	File, GRUB, Kmod, Libelf, Man-DB, og Udev
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, Libelf, Perl, og Util-linux
Valgfrie avhengigheter:	Ingen

Zstd

Installasjonen avhenger av:	Binutils, Coreutils, GCC, Glibc, Gzip, Lz4, Make, Xz, and Zlib
Påkrevd ved kjøretid:	Glibc
Testpakke avhenger av:	Ingen
Må installeres før:	Binutils, GCC, Libelf, and Udev
Valgfrie avhengigheter:	Ingen

Tillegg D. Oppstarts og sysconfig skriptversjon-20240825

Skriptene i dette vedlegget er oppført etter mappene der de er normalt er. Rekkefølgen er `/etc/rc.d/init.d`, `/etc/sysconfig`, `/etc/sysconfig/network-devices`, og `/etc/sysconfig/network-devices/services`. Innenfor hver seksjon vises filene i den rekkefølgen de vanligvis kalles.

D.1. `/etc/rc.d/init.d/rc`

`rc` er det første skriptet som kalles av `init` og starter oppstartsprosessen.

```
#!/bin/bash
#####
# Begin rc
#
# Description : Main Run Level Control Script
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               : DJ Lucas - dj@linuxfromscratch.org
# Updates      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version      : LFS 7.0
#
# Notes        : Updates March 24th, 2022: new semantics of S/K files
#               - Instead of testing that S scripts were K scripts in the
#                 previous runlevel, test that they were not S scripts
#               - Instead of testing that K scripts were S scripts in the
#                 previous runlevel, test that they were not K scripts
#               - S scripts in runlevel 0 or 6 are now run with
#                 "script start" (was "script stop" previously).
#####

. /lib/lsb/init-functions

print_error_msg()
{
    log_failure_msg
    # $i is set when called
    MSG="FAILURE:\n\nYou should not be reading this error message.\n\n"
    MSG="${MSG}It means that an unforeseen error took place in\n"
    MSG="${MSG}${i},\n"
    MSG="${MSG}which exited with a return value of ${error_value}.\n"

    MSG="${MSG}If you're able to track this error down to a bug in one of\n"
    MSG="${MSG}the files provided by the ${DISTRO_MINI} book,\n"
    MSG="${MSG}please be so kind to inform us at ${DISTRO_CONTACT}.\n"
    log_failure_msg "${MSG}"

    log_info_msg "Press Enter to continue..."
    wait_for_user
}

check_script_status()
{
    # $i is set when called
    if [ ! -f ${i} ]; then
        log_warning_msg "${i} is not a valid symlink."
        SCRIPT_STAT="1"
    fi

    if [ ! -x ${i} ]; then
        log_warning_msg "${i} is not executable, skipping."
    fi
}
```

```

    SCRIPT_STAT="1"
    fi
}

run()
{
    if [ -z $interactive ]; then
        ${1} ${2}
        return $?
    fi

    while true; do
        read -p "Run ${1} ${2} (Yes/no/continue)? " -n 1 runit
        echo

        case ${runit} in
            c | C)
                interactive=""
                ${i} ${2}
                ret=${?}
                break;
                ;;

            n | N)
                return 0
                ;;

            y | Y)
                ${i} ${2}
                ret=${?}
                break
                ;;

            *)
                ;;
        esac
    done

    return $ret
}

# Read any local settings/overrides
[ -r /etc/sysconfig/rc.site ] && source /etc/sysconfig/rc.site

DISTRO=${DISTRO:-"Linux From Scratch"}
DISTRO_CONTACT=${DISTRO_CONTACT:-"lfs-dev@lists.linuxfromscratch.org (Registration required)"}
DISTRO_MINI=${DISTRO_MINI:-"LFS"}
IPROMPT=${IPROMPT:-"no"}

# These 3 signals will not cause our script to exit
trap "" INT QUIT TSTP

[ "${1}" != "" ] && runlevel=${1}

if [ "${runlevel}" == "" ]; then
    echo "Usage: ${0} <runlevel>" >&2
    exit 1
fi

previous=${PREVLEVEL}
[ "${previous}" == "" ] && previous=N

if [ ! -d /etc/rc.d/rc${runlevel}.d ]; then
    log_info_msg "/etc/rc.d/rc${runlevel}.d does not exist.\n"
    exit 1
fi

if [ "${runlevel}" == "6" -o "${runlevel}" == "0" ]; then IPROMPT="no"; fi

# Note: In ${LOGLEVEL:-7}, it is ':' 'dash' '7', not minus 7

```

```

if [ "$runlevel" == "S" ]; then
    [ -r /etc/sysconfig/console ] && source /etc/sysconfig/console
    dmesg -n "${LOGLEVEL:-7}"
fi

if [ "${IPROMPT}" == "yes" -a "${runlevel}" == "S" ]; then
    # The total length of the distro welcome string, without escape codes
    wlen=${wlen:-$(echo "Welcome to ${DISTRO}" | wc -c )}
    welcome_message=${welcome_message:-"Welcome to ${INFO}${DISTRO}${NORMAL}"}

    # The total length of the interactive string, without escape codes
    ilen=${ilen:-$(echo "Press 'I' to enter interactive startup" | wc -c )}
    i_message=${i_message:-"Press '${FAILURE}I${NORMAL}' to enter interactive startup"}

    # dcol and icol are spaces before the message to center the message
    # on screen. itime is the amount of wait time for the user to press a key
    wcol=$(( ( ${COLUMNS} - ${wlen} ) / 2 ))
    icol=$(( ( ${COLUMNS} - ${ilen} ) / 2 ))
    itime=${itime:-"3"}

    echo -e "\n\n"
    echo -e "\\033[${wcol}G${welcome_message}"
    echo -e "\\033[${icol}G${i_message}${NORMAL}"
    echo ""
    read -t "${itime}" -n 1 interactive 2>&1 > /dev/null
fi

# Make lower case
[ "${interactive}" == "I" ] && interactive="i"
[ "${interactive}" != "i" ] && interactive=""

# Read the state file if it exists from runlevel S
[ -r /run/interactive ] && source /run/interactive

# Stop all services marked as K, except if marked as K in the previous
# runlevel: it is the responsibility of the script to not try to kill
# a non running service
if [ "${previous}" != "N" ]; then
    for i in $(ls -v /etc/rc.d/rc${runlevel}.d/K* 2> /dev/null)
    do
        check_script_status
        if [ "${SCRIPT_STAT}" == "1" ]; then
            SCRIPT_STAT="0"
            continue
        fi

        suffix=${i#/etc/rc.d/rc${runlevel}.d/K[0-9][0-9]}
        [ -e /etc/rc.d/rc${previous}.d/K[0-9][0-9]$suffix ] && continue

        run ${i} stop
        error_value=${?}

        if [ "${error_value}" != "0" ]; then print_error_msg; fi
    done
fi

if [ "${previous}" == "N" ]; then export IN_BOOT=1; fi

if [ "$runlevel" == "6" -a -n "${FASTBOOT}" ]; then
    touch /fastboot
fi

# Start all services marked as S in this runlevel, except if marked as
# S in the previous runlevel
# it is the responsibility of the script to not try to start an already running

```

```

# service
for i in $( ls -v /etc/rc.d/rc${runlevel}.d/S* 2> /dev/null )
do

    if [ "${previous}" != "N" ]; then
        suffix=${i#/etc/rc.d/rc${runlevel}.d/S[0-9][0-9]}
        [ -e /etc/rc.d/rc${previous}.d/S[0-9][0-9]$suffix ] && continue
    fi

    check_script_status
    if [ "${SCRIPT_STAT}" == "1" ]; then
        SCRIPT_STAT="0"
        continue
    fi

    run ${i} start

    error_value=${?}

    if [ "${error_value}" != "0" ]; then print_error_msg; fi
done

# Store interactive variable on switch from runlevel S and remove if not
if [ "${runlevel}" == "S" -a "${interactive}" == "i" ]; then
    echo "interactive=\"i\"" > /run/interactive
else
    rm -f /run/interactive 2> /dev/null
fi

# Copy the boot log on initial boot only
if [ "${previous}" == "N" -a "${runlevel}" != "S" ]; then
    cat $BOOTLOG >> /var/log/boot.log

    # Mark the end of boot
    echo "-----" >> /var/log/boot.log

    # Remove the temporary file
    rm -f $BOOTLOG 2> /dev/null
fi

# End rc

```

D.2. /lib/lsb/init-functions

```

#!/bin/sh
#####
#
# Begin /lib/lsb/init-funtions
#
# Description : Run Level Control Functions
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#              : DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
# Notes        : With code based on Matthias Benkmann's simpleinit-msb
#              : http://winterdrache.de/linux/newboot/index.html
#
#              The file should be located in /lib/lsb
#
#####

## Environmental setup
# Setup default values for environment
umask 022

```

```

export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

## Set color commands, used via echo
# Please consult `man console_codes` for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font. This does
# not affect framebuffer consoles

NORMAL="\033[0;39m"      # Standard console grey
SUCCESS="\033[1;32m"     # Success is green
WARNING="\033[1;33m"     # Warnings are yellow
FAILURE="\033[1;31m"     # Failures are red
INFO="\033[1;36m"       # Information is light cyan
BRACKET="\033[1;34m"    # Brackets are blue

# Use a colored prefix
BMPREFIX=""
SUCCESS_PREFIX="${SUCCESS} * ${NORMAL} "
FAILURE_PREFIX="${FAILURE} ***** ${NORMAL} "
WARNING_PREFIX="${WARNING} *** ${NORMAL} "
SKIP_PREFIX="${INFO} S ${NORMAL} "

SUCCESS_SUFFIX="${BRACKET}[${SUCCESS} OK ${BRACKET}]${NORMAL}"
FAILURE_SUFFIX="${BRACKET}[${FAILURE} FAIL ${BRACKET}]${NORMAL}"
WARNING_SUFFIX="${BRACKET}[${WARNING} WARN ${BRACKET}]${NORMAL}"
SKIP_SUFFIX="${BRACKET}[${INFO} SKIP ${BRACKET}]${NORMAL}"

BOOTLOG=/run/bootlog
KILLDELAY=3
SCRIPT_STAT="0"

# Set any user specified environment variables e.g. HEADLESS
[ -r /etc/sysconfig/rc.site ] && . /etc/sysconfig/rc.site

# If HEADLESS is set, use that.
# If file descriptor 1 or 2 (stdout and stderr) is not open or
# does not refer to a terminal, consider the script headless.
[ ! -t 1 -o ! -t 2 ] && HEADLESS=${HEADLESS:-yes}

if [ "x$HEADLESS" != "xyes" ]
then
  ## Screen Dimensions
  # Find current screen size
  if [ -z "${COLUMNS}" ]; then
    COLUMNS=$(stty size)
    COLUMNS=${COLUMNS##* }
  fi
else
  COLUMNS=80
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
  COLUMNS=80
fi

## Measurements for positioning result messages
COL=$(( ${COLUMNS} - 8 ))
WCOL=$(( ${COL} - 2 ))

## Set Cursor Position Commands, used via echo
SET_COL="\033[${COL}G"      # at the $COL char
SET_WCOL="\033[${WCOL}G"    # at the $WCOL char
CURS_UP="\033[1A\033[0G"   # Up one line, at the 0'th char

```

```

CURS_ZERO="\033[0G"

#####
# start_daemon() #
# Usage: start_daemon [-f] [-n nicelevel] [-p pidfile] pathname [args...] #
# # #
# Purpose: This runs the specified program as a daemon #
# # #
# Inputs: -f: (force) run the program even if it is already running. #
# -n nicelevel: specify a nice level. See 'man nice(1)'. #
# -p pidfile: use the specified file to determine PIDs. #
# pathname: the complete path to the specified program #
# args: additional arguments passed to the program (pathname) #
# # #
# Return values (as defined by LSB exit codes): #
# 0 - program is running or service is OK #
# 1 - generic or unspecified error #
# 2 - invalid or excessive argument(s) #
# 5 - program is not installed #
#####
start_daemon()
{
    local force=""
    local nice="0"
    local pidfile=""
    local pidlist=""
    local retval=""

    # Process arguments
    while true
    do
        case "${1}" in

            -f)
                force="1"
                shift 1
                ;;

            -n)
                nice="${2}"
                shift 2
                ;;

            -p)
                pidfile="${2}"
                shift 2
                ;;

            -*)
                return 2
                ;;

            *)
                program="${1}"
                break
                ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi

    # Execute
    if [ -z "${force}" ]; then
        if [ -z "${pidfile}" ]; then
            # Determine the pid by discovery
            pidlist=`pidofproc "${1}"`

```

```

    retval="${?}"
else
    # The PID file contains the needed PIDs
    # Note that by LSB requirement, the path must be given to pidofproc,
    # however, it is not used by the current implementation or standard.
    pidlist=`pidofproc -p "${pidfile}" "${1}"`
    retval="${?}"
fi

# Return a value ONLY
# It is the init script's (or distribution's functions) responsibility
# to log messages!
case "${retval}" in

    0)
        # Program is already running correctly, this is a
        # successful start.
        return 0
        ;;

    1)
        # Program is not running, but an invalid pid file exists
        # remove the pid file and continue
        rm -f "${pidfile}"
        ;;

    3)
        # Program is not running and no pidfile exists
        # do nothing here, let start_daemon continue.
        ;;

    *)
        # Others as returned by status values shall not be interpreted
        # and returned as an unspecified error.
        return 1
        ;;

esac

fi

# Do the start!
nice -n "${nice}" "${@"}"
}

#####
# killproc()
# Usage: killproc [-p pidfile] pathname [signal]
#
# Purpose: Send control signals to running processes
#
# Inputs: -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
#         signal, send this signal to pathname
#
# Return values (as defined by LSB exit codes):
#     0 - program (pathname) has stopped/is already stopped or a
#         running program has been sent specified signal and stopped
#         successfully
#     1 - generic or unspecified error
#     2 - invalid or excessive argument(s)
#     5 - program is not installed
#     7 - program is not running and a signal was supplied
#####
killproc()
{
    local pidfile
    local program
    local prefix

```



```

local progname
local signal="-TERM"
local fallback="-KILL"
local nosig
local pidlist
local retval
local pid
local delay="30"
local piddead
local dtime

# Process arguments
while true; do
    case "${1}" in
        -p)
            pidfile="${2}"
            shift 2
            ;;

        *)
            program="${1}"
            if [ -n "${2}" ]; then
                signal="${2}"
                fallback=""
            else
                nosig=1
            fi

            # Error on additional arguments
            if [ -n "${3}" ]; then
                return 2
            else
                break
            fi
            ;;
    esac
done

# Check for a valid program
if [ ! -e "${program}" ]; then return 5; fi

# Check for a valid signal
check_signal "${signal}"
if [ "${?}" -ne "0" ]; then return 2; fi

# Get a list of pids
if [ -z "${pidfile}" ]; then
    # determine the pid by discovery
    pidlist=`pidofproc "${1}"`
    retval="${?}"
else
    # The PID file contains the needed PIDs
    # Note that by LSB requirement, the path must be given to pidofproc,
    # however, it is not used by the current implementation or standard.
    pidlist=`pidofproc -p "${pidfile}" "${1}"`
    retval="${?}"
fi

# Return a value ONLY
# It is the init script's (or distribution's functions) responsibility
# to log messages!
case "${retval}" in

    0)
        # Program is running correctly
        # Do nothing here, let killproc continue.
        ;;

```

```

1)
# Program is not running, but an invalid pid file exists
# Remove the pid file.

progrname=${program##*/}

if [[ -e "/run/${progrname}.pid" ]]; then
    pidfile="/run/${progrname}.pid"
    rm -f "${pidfile}"
fi

# This is only a success if no signal was passed.
if [ -n "${nosig}" ]; then
    return 0
else
    return 7
fi
;;

3)
# Program is not running and no pidfile exists
# This is only a success if no signal was passed.
if [ -n "${nosig}" ]; then
    return 0
else
    return 7
fi
;;

*)
# Others as returned by status values shall not be interpreted
# and returned as an unspecified error.
return 1
;;

esac

# Perform different actions for exit signals and control signals
check_sig_type "${signal}"

if [ "${?}" -eq "0" ]; then # Signal is used to terminate the program

    # Account for empty pidlist (pid file still exists and no
    # signal was given)
    if [ "${pidlist}" != "" ]; then

        # Kill the list of pids
        for pid in ${pidlist}; do

            kill -0 "${pid}" 2> /dev/null

            if [ "${?}" -ne "0" ]; then
                # Process is dead, continue to next and assume all is well
                continue
            else
                kill "${signal}" "${pid}" 2> /dev/null

                # Wait up to ${delay}/10 seconds to for "${pid}" to
                # terminate in 10ths of a second

                while [ "${delay}" -ne "0" ]; do
                    kill -0 "${pid}" 2> /dev/null || piddead="1"
                    if [ "${piddead}" = "1" ]; then break; fi
                    sleep 0.1
                    delay=$(( ${delay} - 1 ))
                done
            fi
        done
    fi
fi

```

```

        # If a fallback is set, and program is still running, then
        # use the fallback
        if [ -n "${fallback}" -a "${piddead}" != "1" ]; then
            kill "${fallback}" "${pid}" 2> /dev/null
            sleep 1
            # Check again, and fail if still running
            kill -0 "${pid}" 2> /dev/null && return 1
        fi
    fi
done
fi

# Check for and remove stale PID files.
if [ -z "${pidfile}" ]; then
    # Find the basename of $program
    prefix=`echo "${program}" | sed 's/[^/]*$//'\`
    proname=`echo "${program}" | sed "s@${prefix}@"\`

    if [ -e "/run/${proname}.pid" ]; then
        rm -f "/run/${proname}.pid" 2> /dev/null
    fi
else
    if [ -e "${pidfile}" ]; then rm -f "${pidfile}" 2> /dev/null; fi
fi

# For signals that do not expect a program to exit, simply
# let kill do its job, and evaluate kill's return for value

else # check_sig_type - signal is not used to terminate program
    for pid in ${pidlist}; do
        kill "${signal}" "${pid}"
        if [ "${?}" -ne "0" ]; then return 1; fi
    done
fi
}

#####
# pidofproc() #
# Usage: pidofproc [-p pidfile] pathname #
# # #
# Purpose: This function returns one or more pid(s) for a particular daemon #
# # #
# Inputs: -p pidfile, use the specified pidfile instead of pidof #
#         pathname, path to the specified program #
# # #
# Return values (as defined by LSB status codes): #
#     0 - Success (PIDs to stdout) #
#     1 - Program is dead, PID file still exists (remaining PIDs output) #
#     3 - Program is not running (no output) #
#####
pidofproc()
{
    local pidfile
    local program
    local prefix
    local proname
    local pidlist
    local lpids
    local exitstatus="0"

    # Process arguments
    while true; do
        case "${1}" in

            -p)
                pidfile="${2}"
                shift 2

```

```

        ;;

        *)
        program="${1}"
        if [ -n "${2}" ]; then
            # Too many arguments
            # Since this is status, return unknown
            return 4
        else
            break
        fi
        ;;
    esac
done

# If a PID file is not specified, try and find one.
if [ -z "${pidfile}" ]; then
    # Get the program's basename
    prefix=`echo "${program}" | sed 's/[^/]*$//`

    if [ -z "${prefix}" ]; then
        procname="${program}"
    else
        procname=`echo "${program}" | sed "s@${prefix}@@"`
    fi

    # If a PID file exists with that name, assume that is it.
    if [ -e "/run/${procname}.pid" ]; then
        pidfile="/run/${procname}.pid"
    fi
fi

# If a PID file is set and exists, use it.
if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
    # Use the value in the first line of the pidfile
    pidlist=`/bin/head -n1 "${pidfile}"`
else
    # Use pidof
    pidlist=`pidof "${program}"`
fi

# Figure out if all listed PIDs are running.
for pid in ${pidlist}; do
    kill -0 ${pid} 2> /dev/null

    if [ "${?}" -eq "0" ]; then
        lpids="${lpids}${pid} "
    else
        exitstatus="1"
    fi
done

if [ -z "${lpids}" -a ! -f "${pidfile}" ]; then
    return 3
else
    echo "${lpids}"
    return "${exitstatus}"
fi
}

#####
# statusproc() #
# Usage: statusproc [-p pidfile] pathname #
# # #
# Purpose: This function prints the status of a particular daemon to stdout #
# # #
# Inputs: -p pidfile, use the specified pidfile instead of pidof #

```

```

#      pathname, path to the specified program      #
#      #      #
# Return values:      #
#      0 - Status printed      #
#      1 - Input error. The daemon to check was not specified.      #
#####
statusproc()
{
    local pidfile
    local pidlist

    if [ "${#}" = "0" ]; then
        echo "Usage: statusproc [-p pidfile] {program}"
        exit 1
    fi

    # Process arguments
    while true; do
        case "${1}" in

            -p)
                pidfile="${2}"
                shift 2
                ;;

            *)
                if [ -n "${2}" ]; then
                    echo "Too many arguments"
                    return 1
                else
                    break
                fi
                ;;
        esac
    done

    if [ -n "${pidfile}" ]; then
        pidlist=`pidofproc -p "${pidfile}" $@`
    else
        pidlist=`pidofproc $@`
    fi

    # Trim trailing blanks
    pidlist=`echo "${pidlist}" | sed -r 's/ +$//`

    base="${1##*/}"

    if [ -n "${pidlist}" ]; then
        /bin/echo -e "${INFO}${base} is running with Process" \
            "ID(s) ${pidlist}.${NORMAL}"
    else
        if [ -n "${base}" -a -e "/run/${base}.pid" ]; then
            /bin/echo -e "${WARNING}${1} is not running but" \
                "/run/${base}.pid exists.${NORMAL}"
        else
            if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
                /bin/echo -e "${WARNING}${1} is not running" \
                    "but ${pidfile} exists.${NORMAL}"
            else
                /bin/echo -e "${INFO}${1} is not running.${NORMAL}"
            fi
        fi
    fi
}

#####
# timespec()      #

```

```

#                                                                                                     #
# Purpose: An internal utility function to format a timestamp                                       #
#           a boot log file.  Sets the STAMP variable.                                             #
#                                                                                                     #
# Return value: Not used                                                                              #
#####
timespec()
{
    STAMP="$(echo `date +%b %d %T %:z` `hostname`) "
    return 0
}

#####
# log_success_msg()                                                                                   #
# Usage: log_success_msg ["message"]                                                                #
#                                                                                                     #
# Purpose: Print a successful status message to the screen and                                     #
#           a boot log file.                                                                        #
#                                                                                                     #
# Inputs:  $@ - Message                                                                              #
#                                                                                                     #
# Return values: Not used                                                                            #
#####
log_success_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
        /bin/echo -n -e "${BMPREFIX}${@}"
        /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"
    else
        logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
        /bin/echo -e "${logmessage} OK"
    fi
    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`

    timespec
    /bin/echo -e "${STAMP} ${logmessage} OK" >> ${BOOTLOG}

    return 0
}

log_success_msg2()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
        /bin/echo -n -e "${BMPREFIX}${@}"
        /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"
    else
        echo " OK"
    fi

    echo " OK" >> ${BOOTLOG}

    return 0
}

#####
# log_failure_msg()                                                                                   #
# Usage: log_failure_msg ["message"]                                                                #
#                                                                                                     #
# Purpose: Print a failure status message to the screen and                                       #
#           a boot log file.                                                                        #
#                                                                                                     #
# Inputs:  $@ - Message                                                                              #
#                                                                                                     #
# Return values: Not used                                                                            #

```

```
#####
log_failure_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
        /bin/echo -n -e "${BMPREFIX}${@}"
        /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"
    else
        logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
        /bin/echo -e "${logmessage} FAIL"
    fi

    # Strip non-printable characters from log file

    timespec
    logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
    /bin/echo -e "${STAMP} ${logmessage} FAIL" >> ${BOOTLOG}

    return 0
}

log_failure_msg2()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
        /bin/echo -n -e "${BMPREFIX}${@}"
        /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"
    else
        echo "FAIL"
    fi

    echo "FAIL" >> ${BOOTLOG}

    return 0
}

#####
# log_warning_msg() #
# Usage: log_warning_msg ["message"] #
# # #
# Purpose: Print a warning status message to the screen and #
# a boot log file. #
# # #
# Return values: Not used #
#####
log_warning_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
        /bin/echo -n -e "${BMPREFIX}${@}"
        /bin/echo -e "${CURS_ZERO}${WARNING_PREFIX}${SET_COL}${WARNING_SUFFIX}"
    else
        logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
        /bin/echo -e "${logmessage} WARN"
    fi

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
    timespec
    /bin/echo -e "${STAMP} ${logmessage} WARN" >> ${BOOTLOG}

    return 0
}

log_skip_msg()
{
    if [ "x$HEADLESS" != "xyes" ]

```

```

then
  /bin/echo -n -e "${BMPREFIX}${@}"
  /bin/echo -e "${CURS_ZERO}${SKIP_PREFIX}${SET_COL}${SKIP_SUFFIX}"
else
  logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
  /bin/echo "SKIP"
fi

# Strip non-printable characters from log file
logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
/bin/echo "SKIP" >> ${BOOTLOG}

return 0
}

#####
# log_info_msg()                                     #
# Usage: log_info_msg message                       #
#                                                  #
# Purpose: Print an information message to the screen and #
#          a boot log file. Does not print a trailing newline character. #
#                                                  #
# Return values: Not used                           #
#####
log_info_msg()
{
  if [ "x$HEADLESS" != "xyes" ]
  then
    /bin/echo -n -e "${BMPREFIX}${@}"
  else
    logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
    /bin/echo -n -e "${logmessage}"
  fi

  # Strip non-printable characters from log file
  logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
  timespec
  /bin/echo -n -e "${STAMP} ${logmessage}" >> ${BOOTLOG}

  return 0
}

log_info_msg2()
{
  if [ "x$HEADLESS" != "xyes" ]
  then
    /bin/echo -n -e "${@}"
  else
    logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
    /bin/echo -n -e "${logmessage}"
  fi

  # Strip non-printable characters from log file
  logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
  /bin/echo -n -e "${logmessage}" >> ${BOOTLOG}

  return 0
}

#####
# evaluate_retval()                                   #
# Usage: Evaluate a return value and print success or failure as appropriate #
#                                                  #
# Purpose: Convenience function to terminate an info message #
#                                                  #
# Return values: Not used                           #
#####

```



```

evaluate_retval()
{
    local error_value="${?}"

    if [ ${error_value} = 0 ]; then
        log_success_msg2
    else
        log_failure_msg2
    fi
}

#####
# check_signal()
# Usage: check_signal [ -{signal} ]
#
# Purpose: Check for a valid signal. This is not defined by any LSB draft,
#         however, it is required to check the signals to determine if the
#         signals chosen are invalid arguments to the other functions.
#
# Inputs: Accepts a single string value in the form of -{signal}
#
# Return values:
#     0 - Success (signal is valid)
#     1 - Signal is not valid
#####
check_signal()
{
    local valsig

    # Add error handling for invalid signals
    valsig="-ALRM -HUP -INT -KILL -PIPE -POLL -PROF -TERM -USR1 -USR2"
    valsig="${valsig} -VTALRM -STKFLT -PWR -WINCH -CHLD -URG -TSTP -TTIN"
    valsig="${valsig} -TTOU -STOP -CONT -ABRT -FPE -ILL -QUIT -SEGV -TRAP"
    valsig="${valsig} -SYS -EMT -BUS -XCPU -XFSZ -0 -1 -2 -3 -4 -5 -6 -8 -9"
    valsig="${valsig} -11 -13 -14 -15 "

    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}

#####
# check_sig_type()
# Usage: check_signal [ -{signal} | {signal} ]
#
# Purpose: Check if signal is a program termination signal or a control signal
#         This is not defined by any LSB draft, however, it is required to
#         check the signals to determine if they are intended to end a
#         program or simply to control it.
#
# Inputs: Accepts a single string value in the form or -{signal} or {signal}
#
# Return values:
#     0 - Signal is used for program termination
#     1 - Signal is used for program control
#####
check_sig_type()
{
    local valsig

    # The list of termination signals (limited to generally used items)
    valsig="-ALRM -INT -KILL -TERM -PWR -STOP -ABRT -QUIT -2 -3 -6 -9 -14 -15 "

```

```

echo "${valsig}" | grep -- " ${1} " > /dev/null

if [ "${?}" -eq "0" ]; then
    return 0
else
    return 1
fi
}

#####
# wait_for_user()                                     #
#                                                     #
# Purpose: Wait for the user to respond if not a headless system #
#                                                     #
#####
wait_for_user()
{
    # Wait for the user by default
    [ "${HEADLESS=0}" = "0" ] && read ENTER
    return 0
}

#####
# is_true()                                           #
#                                                     #
# Purpose: Utility to test if a variable is true | yes | 1 #
#                                                     #
#####
is_true()
{
    [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ] || [ "$1" = "y" ] ||
    [ "$1" = "t" ]
}

# End /lib/lsb/init-functions

```

D.3. /etc/rc.d/init.d/mountvirtfs

```

#!/bin/sh
#####
# Begin mountvirtfs
#
# Description : Ensure proc, sysfs, run, and dev are mounted
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Xi Ruoyao - xry111@xry111.site
#
# Version      : LFS 12.0
#
#####

### BEGIN INIT INFO
# Provides:          mountvirtfs
# Required-Start:    $first
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Mounts various special fs needed at start
# Description:       Mounts /sys and /proc virtual (kernel) filesystems.
#                   Mounts /run (tmpfs) and /dev (devtmpfs).
#                   This is done only if they are not already mounted.
#                   with the kernel config proposed in the book, dev
#                   should be automatically mounted by the kernel.
#

```

```

# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
  start)
    # Make sure /run is available before logging any messages
    if ! mountpoint /run >/dev/null; then
      mount /run || failed=1
    fi

    mkdir -p /run/lock
    chmod 1777 /run/lock

    log_info_msg "Mounting virtual file systems: ${INFO}/run"

    if ! mountpoint /proc >/dev/null; then
      log_info_msg2 " ${INFO}/proc"
      mount -o nosuid,noexec,nodev /proc || failed=1
    fi

    if ! mountpoint /sys >/dev/null; then
      log_info_msg2 " ${INFO}/sys"
      mount -o nosuid,noexec,nodev /sys || failed=1
    fi

    if ! mountpoint /dev >/dev/null; then
      log_info_msg2 " ${INFO}/dev"
      mount -o mode=0755,nosuid /dev || failed=1
    fi

    mkdir -p /dev/shm
    log_info_msg2 " ${INFO}/dev/shm"
    mount -o nosuid,nodev /dev/shm || failed=1

    mkdir -p /sys/fs/cgroup
    log_info_msg2 " ${INFO}/sys/fs/cgroup"
    mount -o nosuid,noexec,nodev /sys/fs/cgroup || failed=1

    (exit ${failed})
    evaluate_retval
    if [ "${failed}" = 1 ]; then
      exit 1
    fi

    log_info_msg "Create symlinks in /dev targeting /proc: ${INFO}/dev/stdin"
    ln -sf /proc/self/fd/0 /dev/stdin || failed=1

    log_info_msg2 " ${INFO}/dev/stdout"
    ln -sf /proc/self/fd/1 /dev/stdout || failed=1

    log_info_msg2 " ${INFO}/dev/stderr"
    ln -sf /proc/self/fd/2 /dev/stderr || failed=1

    log_info_msg2 " ${INFO}/dev/fd"
    ln -sf /proc/self/fd /dev/fd || failed=1

    if [ -e /proc/kcore ]; then
      log_info_msg2 " ${INFO}/dev/core"
      ln -sf /proc/kcore /dev/core || failed=1
    fi

    (exit ${failed})
    evaluate_retval
    exit $failed
  ;;

```

```

*)
    echo "Usage: ${0} {start}"
    exit 1
    ;;
esac

# End mountvirtfs

```

D.4. /etc/rc.d/init.d/modules

```

#!/bin/sh
#####
# Begin modules
#
# Description : Module auto-loading script
#
# Authors      : Zack Winkles
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:      modules
# Required-Start: mountvirtfs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start: S
# Default-Stop:
# Short-Description: Loads required modules.
# Description:     Loads modules listed in /etc/sysconfig/modules.
# X-LFS-Provided-By: LFS
### END INIT INFO

# Assure that the kernel has module support.
[ -e /proc/modules ] || exit 0

. /lib/lsb/init-functions

case "${1}" in
    start)
        # Exit if there's no modules file or there are no
        # valid entries
        [ -r /etc/sysconfig/modules ] || exit 0
        grep -E -qv '^(#|)' /etc/sysconfig/modules || exit 0

        log_info_msg "Loading modules:"

        # Only try to load modules if the user has actually given us
        # some modules to load.

        while read module args; do

            # Ignore comments and blank lines.
            case "$module" in
                ""|"#") continue ;;
            esac

            # Attempt to load the module, passing any arguments provided.
            modprobe ${module} ${args} >/dev/null

            # Print the module name if successful, otherwise take note.
            if [ $? -eq 0 ]; then

```

```

        log_info_msg2 " ${module}"
    else
        failedmod="${failedmod} ${module}"
    fi
done < /etc/sysconfig/modules

# Print a message about successfully loaded modules on the correct line.
log_success_msg2

# Print a failure message with a list of any modules that
# may have failed to load.
if [ -n "${failedmod}" ]; then
    log_failure_msg "Failed to load modules:${failedmod}"
    exit 1
fi
;;

*)
    echo "Usage: ${0} {start}"
    exit 1
    ;;
esac

exit 0

# End modules

```

D.5. /etc/rc.d/init.d/udev

```

#!/bin/sh
#####
# Begin udev
#
# Description : Udev cold-plugging script
#
# Authors      : Zack Winkles, Alexander E. Patrakov
#                DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#                Xi Ruoyao - xry111@xry111.site
#
# Version      : LFS 12.0
#
#####

### BEGIN INIT INFO
# Provides:          udev $time
# Required-Start:    localnet
# Should-Start:      modules
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Populates /dev with device nodes.
# Description:       Mounts a tempfs on /dev and starts the udevd daemon.
#                    Device nodes are created as defined by udev.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Populating /dev with device nodes... "
        if ! grep -q '[:space:]sysfs' /proc/mounts; then
            log_failure_msg2
            msg="FAILURE:\n\nUnable to create "
            msg="${msg}devices without a SysFS filesystem\n\n"

```

```

    msg="${msg}After you press Enter, this system "
    msg="${msg}will be halted and powered off.\n\n"
    log_info_msg "$msg"
    log_info_msg "Press Enter to continue..."
    wait_for_user
    /etc/rc.d/init.d/halt start
fi

# Start the udev daemon to continually watch for, and act on,
# uevents
SYSTEMD_LOG_TARGET=kmsg /sbin/udev --daemon

# Now traverse /sys in order to "coldplug" devices that have
# already been discovered
/bin/udevadm trigger --action=add    --type=subsystems
/bin/udevadm trigger --action=add    --type=devices
/bin/udevadm trigger --action=change --type=devices

# Now wait for udevd to process the uevents we triggered
if ! is_true "$OMIT_UDEV_SETTLE"; then
    /sbin/udevadm settle
fi

# If any LVM based partitions are on the system, ensure they
# are activated so they can be used.
if [ -x /sbin/vgchange ]; then /sbin/vgchange -a y >/dev/null; fi

log_success_msg2
;;

*)
    echo "Usage ${0} {start}"
    exit 1
;;

esac

exit 0

# End udev

```

D.6. /etc/rc.d/init.d/swap

```

#!/bin/sh
#####
# Begin swap
#
# Description : Swap Control Script
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:      swap
# Required-Start: udev
# Should-Start:  modules
# Required-Stop: localnet
# Should-Stop:   $local_fs
# Default-Start: S
# Default-Stop:  0 6
# Short-Description: Activates and deactivates swap partitions.
# Description:     Activates and deactivates swap partitions defined in
#                  /etc/fstab.
#

```

```

# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Activating all swap files/partitions..."
        swapon -a
        evaluate_retval
        ;;

    stop)
        log_info_msg "Deactivating all swap files/partitions..."
        swapoff -a
        evaluate_retval
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    status)
        log_success_msg "Retrieving swap status."
        swapon -s
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart|status}"
        exit 1
        ;;
esac

exit 0

# End swap

```

D.7. /etc/rc.d/init.d/setclock

```

#!/bin/sh
#####
# Begin setclock
#
# Description : Setting Linux Clock
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:
# Required-Start:
# Should-Start:      modules
# Required-Stop:
# Should-Stop:       $syslog
# Default-Start:     S
# Default-Stop:
# Short-Description: Stores and restores time from the hardware clock
# Description:        On boot, system time is obtained from hwclock. The
#                     hardware clock can also be set on shutdown.
#
# X-LFS-Provided-By: LFS

```

```

### END INIT INFO

. /lib/lsb/init-functions

[ -r /etc/sysconfig/clock ] && . /etc/sysconfig/clock

case "${UTC}" in
    yes|true|1)
        CLOCKPARAMS="${CLOCKPARAMS} --utc"
        ;;

    no|false|0)
        CLOCKPARAMS="${CLOCKPARAMS} --localtime"
        ;;

esac

case ${1} in
    start)
        hwclock --hctosys ${CLOCKPARAMS} >/dev/null
        ;;

    stop)
        log_info_msg "Setting hardware clock..."
        hwclock --systohc ${CLOCKPARAMS} >/dev/null
        evaluate_retval
        ;;

    *)
        echo "Usage: ${0} {start|stop}"
        exit 1
        ;;

esac

exit 0

```

D.8. /etc/rc.d/init.d/checkfs

```

#!/bin/sh
#####
# Begin checkfs
#
# Description : File System Check
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               A. Luebke - luebke@users.sourceforge.net
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
# Based on checkfs script from LFS-3.1 and earlier.
#
# From man fsck
# 0      - No errors
# 1      - File system errors corrected
# 2      - System should be rebooted
# 4      - File system errors left uncorrected
# 8      - Operational error
# 16     - Usage or syntax error
# 32     - Fsck canceled by user request
# 128    - Shared library error
#
#####

```



```

### BEGIN INIT INFO
# Provides:          checkfs
# Required-Start:    udev swap
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Checks local filesystems before mounting.
# Description:       Checks local filesystems before mounting.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        if [ -f /fastboot ]; then
            msg="/fastboot found, will omit "
            msg="${msg} file system checks as requested.\n"
            log_info_msg "${msg}"
            exit 0
        fi

        log_info_msg "Mounting root file system in read-only mode... "
        mount -n -o remount,ro / >/dev/null

        if [ ${?} != 0 ]; then
            log_failure_msg2
            msg="\n\nCannot check root "
            msg="${msg}filesystem because it could not be mounted "
            msg="${msg}in read-only mode.\n\n"
            msg="${msg}After you press Enter, this system will be "
            msg="${msg}halted and powered off.\n\n"
            log_failure_msg "${msg}"

            log_info_msg "Press Enter to continue..."
            wait_for_user
            /etc/rc.d/init.d/halt start
        else
            log_success_msg2
        fi

        if [ -f /forcefsck ]; then
            msg="/forcefsck found, forcing file"
            msg="${msg} system checks as requested."
            log_success_msg "${msg}"
            options="-f"
        else
            options=""
        fi

        log_info_msg "Checking file systems..."
        # Note: -a option used to be -p; but this fails e.g. on fsck.minix
        if is_true "$VERBOSE_FSCK"; then
            fsck ${options} -a -A -C -T
        else
            fsck ${options} -a -A -C -T >/dev/null
        fi

        error_value=${?}

        if [ "${error_value}" = 0 ]; then
            log_success_msg2
        fi

        if [ "${error_value}" = 1 ]; then

```

```

msg="\nWARNING:\n\nFile system errors "
msg="${msg}were found and have been corrected.\n"
msg="${msg}      You may want to double-check that "
msg="${msg}everything was fixed properly."
log_warning_msg "$msg"
fi

if [ "${error_value}" = 2 -o "${error_value}" = 3 ]; then
msg="\nWARNING:\n\nFile system errors "
msg="${msg}were found and have been "
msg="${msg}corrected, but the nature of the "
msg="${msg}errors require this system to be rebooted.\n\n"
msg="${msg}After you press enter, "
msg="${msg}this system will be rebooted\n\n"
log_failure_msg "$msg"

log_info_msg "Press Enter to continue..."
wait_for_user
reboot -f
fi

if [ "${error_value}" -gt 3 -a "${error_value}" -lt 16 ]; then
msg="\nFAILURE:\n\nFile system errors "
msg="${msg}were encountered that could not be "
msg="${msg}fixed automatically.\nThis system "
msg="${msg}cannot continue to boot and will "
msg="${msg}therefore be halted until those "
msg="${msg}errors are fixed manually by a "
msg="${msg}System Administrator.\n\n"
msg="${msg}After you press Enter, this system will be "
msg="${msg}halted and powered off.\n\n"
log_failure_msg "$msg"

log_info_msg "Press Enter to continue..."
wait_for_user
/etc/rc.d/init.d/halt start
fi

if [ "${error_value}" -ge 16 ]; then
msg="FAILURE:\n\nUnexpected failure "
msg="${msg}running fsck. Exited with error "
msg="${msg} code: ${error_value}.\n"
log_info_msg $msg
exit ${error_value}
fi

exit 0
;;
*)
echo "Usage: ${0} {start}"
exit 1
;;
esac

# End checkfs

```

D.9. /etc/rc.d/init.d/mountfs

```

#!/bin/sh
#####
# Begin mountfs
#
# Description : File System Mount Script
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#              : DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org

```

```

#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:    $local_fs
# Required-Start:    udev checkfs
# Should-Start:    modules
# Required-Stop:    localnet
# Should-Stop:
# Default-Start:    S
# Default-Stop:    0 6
# Short-Description:    Mounts/unmounts local filesystems defined in /etc/fstab.
# Description:    Remounts root filesystem read/write and mounts all
#                  remaining local filesystems defined in /etc/fstab on
#                  start. Remounts root filesystem read-only and unmounts
#                  remaining filesystems on stop.
# X-LFS-Provided-By:    LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Remounting root file system in read-write mode..."
        mount --options remount,rw / >/dev/null
        evaluate_retval

        # Remove fsck-related file system watermarks.
        rm -f /fastboot /forcefsck

        # Make sure /dev/pts exists
        mkdir -p /dev/pts

        # This will mount all filesystems that do not have _netdev in
        # their option list. _netdev denotes a network filesystem.

        log_info_msg "Mounting remaining file systems..."
        failed=0
        mount --all --test-opts no_netdev >/dev/null || failed=1
        evaluate_retval
        exit $failed
        ;;

    stop)
        # Don't unmount virtual file systems like /run
        log_info_msg "Unmounting all other currently mounted file systems..."
        # Ensure any loop devices are removed
        losetup -D
        umount --all --detach-loop --read-only \
            --types notmpfs,nosysfs,nodevtmpfs,noproc,nodevpts >/dev/null
        evaluate_retval

        # Make sure / is mounted read only (umount bug)
        mount --options remount,ro /

        # Make all LVM volume groups unavailable, if appropriate
        # This fails if swap or / are on an LVM partition
        #if [ -x /sbin/vgchange ]; then /sbin/vgchange -an > /dev/null; fi
        if [ -r /etc/mdadm.conf ]; then
            log_info_msg "Mark arrays as clean..."
            mdadm --wait-clean --scan
            evaluate_retval
        fi
        ;;
)

```

```

*)
    echo "Usage: ${0} {start|stop}"
    exit 1
    ;;
esac

# End mountfs

```

D.10. /etc/rc.d/init.d/udev_retry

```

#!/bin/sh
#####
# Begin udev_retry
#
# Description : Udev cold-plugging script (retry)
#
# Authors      : Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Bryan Kadzban -
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:          udev_retry
# Required-Start:    udev
# Should-Start:      $local_fs cleanfs
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Replays failed uevents and creates additional devices.
# Description:       Replays any failed uevents that were skipped due to
#                   slow hardware initialization, and creates those needed
#                   device nodes
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Retrying failed uevents, if any..."

        rundir=/run/udev
        # From Debian: "copy the rules generated before / was mounted
        # read-write":

        for file in ${rundir}/tmp-rules--*; do
            dest=${file##*tmp-rules--}
            [ "$dest" = '*' ] && break
            cat $file >> /etc/udev/rules.d/$dest
            rm -f $file
        done

        # Re-trigger the uevents that may have failed,
        # in hope they will succeed now
        /bin/sed -e 's/#.*$//' /etc/sysconfig/udev_retry | /bin/grep -v '^$' | \
        while read line ; do
            for subsystem in $line ; do
                /bin/udevadm trigger --subsystem-match=$subsystem --action=add
            done
        done

        # Now wait for udevd to process the uevents we triggered

```

```

    if ! is_true "$OMIT_UDEV_RETRY_SETTLE"; then
        /bin/udevadm settle
    fi

    evaluate_retval
    ;;

*)
    echo "Usage ${0} {start}"
    exit 1
    ;;
esac

exit 0

# End udev_retry

```

D.11. /etc/rc.d/init.d/cleanfs

```

#!/bin/sh
#####
# Begin cleanfs
#
# Description : Clean file system
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:          cleanfs
# Required-Start:    $local_fs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Cleans temporary directories early in the boot process.
# Description:       Cleans temporary directories /run, /var/lock, and
#                   optionally, /tmp. cleanfs also creates /run/utmp
#                   and any files defined in /etc/sysconfig/createfiles.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

# Function to create files/directory on boot.
create_files()
{
    # Input to file descriptor 9 and output to stdin (redirection)
    exec 9>&0 < /etc/sysconfig/createfiles

    while read name type perm usr grp dtype maj min junk
    do
        # Ignore comments and blank lines.
        case "${name}" in
            ""|\#*) continue ;;
        esac

        # Ignore existing files.
        if [ ! -e "${name}" ]; then
            # Create stuff based on its type.
            case "${type}" in

```

```

    dir)
        mkdir "${name}"
        ;;
    file)
        :> "${name}"
        ;;
    dev)
        case "${dtype}" in
            char)
                mknod "${name}" c ${maj} ${min}
                ;;
            block)
                mknod "${name}" b ${maj} ${min}
                ;;
            pipe)
                mknod "${name}" p
                ;;
            *)
                log_warning_msg "\nUnknown device type: ${dtype}"
                ;;
        esac
        ;;
    *)
        log_warning_msg "\nUnknown type: ${type}"
        continue
        ;;
esac

# Set up the permissions, too.
chown ${usr}:${grp} "${name}"
chmod ${perm} "${name}"
fi
done

# Close file descriptor 9 (end redirection)
exec 0>&9 9>&-
return 0
}

case "${1}" in
    start)
        log_info_msg "Cleaning file systems:"

        if [ "${SKIPTMPCLEAN}" = "" ]; then
            log_info_msg2 " /tmp"
            cd /tmp &&
            find . -xdev -mindepth 1 ! -name lost+found -delete || failed=1
        fi

        > /run/utmp

        if grep -q '^utmp:' /etc/group ; then
            chmod 664 /run/utmp
            chgrp utmp /run/utmp
        fi

        (exit ${failed})
        evaluate_retval

        if grep -E -qv '^(#|$)' /etc/sysconfig/createfiles 2>/dev/null; then
            log_info_msg "Creating files and directories... "
            create_files      # Always returns 0
            evaluate_retval
        fi

        exit $failed
        ;;

```

```

*)
    echo "Usage: ${0} {start}"
    exit 1
    ;;
esac

# End cleanfs

```

D.12. /etc/rc.d/init.d/console

```

#!/bin/sh
#####
# Begin console
#
# Description : Sets keymap and screen font
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:      console
# Required-Start: $local_fs
# Should-Start:  udev_retry
# Required-Stop:
# Should-Stop:
# Default-Start: S
# Default-Stop:
# Short-Description: Sets up a localised console.
# Description:     Sets up fonts and language settings for the user's
#                  local as defined by /etc/sysconfig/console.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

# Native English speakers probably don't have /etc/sysconfig/console at all
[ -r /etc/sysconfig/console ] && . /etc/sysconfig/console

failed=0

case "${1}" in
    start)
        # See if we need to do anything
        if [ -z "${KEYMAP}" ] && [ -z "${KEYMAP_CORRECTIONS}" ] &&
            [ -z "${FONT}" ] && [ -z "${LEGACY_CHARSET}" ] &&
            ! is_true "${UNICODE}"; then
            exit 0
        fi

        # There should be no bogus failures below this line!
        log_info_msg "Setting up Linux console..."

        # Figure out if a framebuffer console is used
        [ -d /sys/class/graphics/fbcon ] && use_fb=1 || use_fb=0

        # Figure out the command to set the console into the
        # desired mode
        is_true "${UNICODE}" &&
            MODE_COMMAND="echo -en '\033%G' && kbd_mode -u" ||
            MODE_COMMAND="echo -en '\033%@\033(K' && kbd_mode -a"
    esac

```

```

# On framebuffer consoles, font has to be set for each vt in
# UTF-8 mode. This doesn't hurt in non-UTF-8 mode also.

! is_true "${use_fb}" || [ -z "${FONT}" ] ||
  MODE_COMMAND="${MODE_COMMAND} && setfont ${FONT}"

# Apply that command to all consoles mentioned in
# /etc/inittab. Important: in the UTF-8 mode this should
# happen before setfont, otherwise a kernel bug will
# show up and the unicode map of the font will not be
# used.

for TTY in `grep '^[^#].*respawn:/sbin/agetty' /etc/inittab |
  grep -o '\btty[[:digit:]]*\b'`
do
  openvt -f -w -c ${TTY#tty} -- \
    /bin/sh -c "${MODE_COMMAND}" || failed=1
done

# Set the font (if not already set above) and the keymap
[ "${use_fb}" == "1" ] || [ -z "${FONT}" ] || setfont $FONT || failed=1

[ -z "${KEYMAP}" ] ||
  loadkeys ${KEYMAP} >/dev/null 2>&1 ||
  failed=1

[ -z "${KEYMAP_CORRECTIONS}" ] ||
  loadkeys ${KEYMAP_CORRECTIONS} >/dev/null 2>&1 ||
  failed=1

# Convert the keymap from $LEGACY_CHARSET to UTF-8
[ -z "${LEGACY_CHARSET}" ] ||
  dumpkeys -c "${LEGACY_CHARSET}" | loadkeys -u >/dev/null 2>&1 ||
  failed=1

# If any of the commands above failed, the trap at the
# top would set $failed to 1
( exit $failed )
evaluate_retval

exit $failed
;;

*)
  echo "Usage:  ${0} {start}"
  exit 1
  ;;
esac

# End console

```

D.13. /etc/rc.d/init.d/localnet

```

#!/bin/sh
#####
# Begin localnet
#
# Description : Loopback device
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#              : DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
#####

```



```

### BEGIN INIT INFO
# Provides:          localnet
# Required-Start:   mountvirtfs
# Should-Start:    modules
# Required-Stop:
# Should-Stop:
# Default-Start:   S
# Default-Stop:    0 6
# Short-Description: Starts the local network.
# Description:     Sets the hostname of the machine and starts the
#                  loopback interface.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions
[ -r /etc/sysconfig/network ] && . /etc/sysconfig/network
[ -r /etc/hostname ] && HOSTNAME=`cat /etc/hostname`

case "${1}" in
    start)
        log_info_msg "Bringing up the loopback interface..."
        ip addr add 127.0.0.1/8 label lo dev lo
        ip link set lo up
        evaluate_retval

        log_info_msg "Setting hostname to ${HOSTNAME}..."
        hostname ${HOSTNAME}
        evaluate_retval
        ;;

    stop)
        log_info_msg "Bringing down the loopback interface..."
        ip link set lo down
        evaluate_retval
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    status)
        echo "Hostname is: $(hostname)"
        ip link show lo
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart|status}"
        exit 1
        ;;
esac

exit 0

# End localnet

```

D.14. /etc/rc.d/init.d/sysctl

```

#!/bin/sh
#####
# Begin sysctl
#
# Description : File uses /etc/sysctl.conf to set kernel runtime
#              parameters
#
# Authors      : Nathan Coulson (nathan@linuxfromscratch.org)

```

```

#           Matthew Burgess (matthew@linuxfromscratch.org)
#           DJ Lucas - dj@linuxfromscratch.org
# Update    : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version   : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:          sysctl
# Required-Start:    mountvirtfs
# Should-Start:      console
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Makes changes to the proc filesystem
# Description:        Makes changes to the proc filesystem as defined in
#                     /etc/sysctl.conf.  See 'man sysctl(8)'.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        if [ -f "/etc/sysctl.conf" ]; then
            log_info_msg "Setting kernel runtime parameters..."
            sysctl -q -p
            evaluate_retval
        fi
        ;;

    status)
        sysctl -a
        ;;

    *)
        echo "Usage: ${0} {start|status}"
        exit 1
        ;;
esac

exit 0

# End sysctl

```

D.15. /etc/rc.d/init.d/sysklogd

```

#!/bin/sh
#####
# Begin sysklogd
#
# Description : Sysklogd loader
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#             DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org LFS12.1
#             Remove kernel log daemon.  The functionality has been
#             merged with syslogd.
#
# Version     : LFS 7.0
#
#####

### BEGIN INIT INFO

```

```

# Provides:          $syslog
# Required-Start:    $first localnet
# Should-Start:
# Required-Stop:     $local_fs
# Should-Stop:       sendsignals
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Starts system log daemon.
# Description:       Starts system log daemon.
#                   /etc/fstab.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Starting system log daemon..."
        parms=${SYSKLOGD_PARMS-'-m 0'}
        start_daemon /sbin/syslogd $parms
        evaluate_retval
        ;;

    stop)
        log_info_msg "Stopping system log daemon..."
        killproc /sbin/syslogd
        evaluate_retval
        ;;

    reload)
        log_info_msg "Reloading system log daemon config file..."
        pid=`pidofproc syslogd`
        kill -HUP "${pid}"
        evaluate_retval
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    status)
        statusproc /sbin/syslogd
        ;;

    *)
        echo "Usage: ${0} {start|stop|reload|restart|status}"
        exit 1
        ;;
esac

exit 0

# End syslogd

```

D.16. /etc/rc.d/init.d/network

```

#!/bin/sh
#####
# Begin network
#
# Description : Network Control Script
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org

```

```

#           DJ Lucas - dj@linuxfromscratch.org
# Update    : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version   : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:          $network
# Required-Start:    $local_fs localnet swap
# Should-Start:     $syslog firewalld iptables nftables
# Required-Stop:    $local_fs localnet swap
# Should-Stop:      $syslog firewalld iptables nftables
# Default-Start:    2 3 4 5
# Default-Stop:     0 1 6
# Short-Description: Starts and configures network interfaces.
# Description:       Starts and configures network interfaces.
# X-LFS-Provided-By: LFS
### END INIT INFO

case "${1}" in
    start)
        # if the default route exists, network is already configured
        if ip route | grep -q "^default"; then return 0; fi
        # Start all network interfaces
        for file in /etc/sysconfig/ifconfig.*
        do
            interface=${file##*/ifconfig.}

            # Skip if $file is * (because nothing was found)
            if [ "${interface}" = "*" ]; then continue; fi

            /sbin/ifup ${interface}
        done
        ;;

    stop)
        # Unmount any network mounted file systems
        umount --all --force --types nfs,cifs,nfs4

        # Reverse list
        net_files=""
        for file in /etc/sysconfig/ifconfig.*
        do
            net_files="${file} ${net_files}"
        done

        # Stop all network interfaces
        for file in ${net_files}
        do
            interface=${file##*/ifconfig.}

            # Skip if $file is * (because nothing was found)
            if [ "${interface}" = "*" ]; then continue; fi

            # See if interface exists
            if [ ! -e /sys/class/net/${interface} ]; then continue; fi

            # Is interface UP?
            ip link show $interface 2>/dev/null | grep -q "state UP"
            if [ $? -ne 0 ]; then continue; fi

            /sbin/ifdown ${interface}
        done
        ;;

    restart)

```

```

    ${0} stop
    sleep 1
    ${0} start
    ;;

*)
    echo "Usage: ${0} {start|stop|restart}"
    exit 1
    ;;
esac

exit 0

# End network

```

D.17. /etc/rc.d/init.d/sendsignals

```

#!/bin/sh
#####
# Begin sendsignals
#
# Description : Sendsignals Script
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:          sendsignals
# Required-Start:
# Should-Start:
# Required-Stop:     $local_fs swap localnet
# Should-Stop:
# Default-Start:
# Default-Stop:      0 6
# Short-Description: Attempts to kill remaining processes.
# Description:       Attempts to kill remaining processes.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    stop)
        omit=$(pidof mdmon)
        [ -n "$omit" ] && omit="-o $omit"

        log_info_msg "Sending all processes the TERM signal..."
        killall15 -15 $omit
        error_value=${?}

        sleep ${KILLDELAY}

        if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
            log_success_msg
        else
            log_failure_msg
        fi

        log_info_msg "Sending all processes the KILL signal..."
        killall15 -9 $omit
        error_value=${?}

```

```

    sleep ${KILLDELAY}

    if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
        log_success_msg
    else
        log_failure_msg
    fi
    ;;

*)
    echo "Usage: ${0} {stop}"
    exit 1
    ;;

esac

exit 0

# End sendsignals

```

D.18. /etc/rc.d/init.d/reboot

```

#!/bin/sh
#####
# Begin reboot
#
# Description : Reboot Scripts
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Updates      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Pierre Labastie - pierre@linuxfromscratch.org
#
# Version      : LFS 7.0
#
# Notes        : Update March 24th, 2022: change "stop" to "start".
#               Add the $last facility to Required-start
#
#####

### BEGIN INIT INFO
# Provides:          reboot
# Required-Start:    $last
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:     6
# Default-Stop:
# Short-Description: Reboots the system.
# Description:       Reboots the System.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Restarting system..."
        reboot -d -f -i
        ;;

    *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;

esac

```

```
# End reboot
```

D.19. /etc/rc.d/init.d/halt

```
#!/bin/sh
#####
# Begin halt
#
# Description : Halt Script
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version      : LFS 7.0
#
# Notes        : Update March 24th, 2022: change "stop" to "start".
#               Add the $last facility to Required-start
#
#####

### BEGIN INIT INFO
# Provides:          halt
# Required-Start:    $last
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:    0
# Default-Stop:
# Short-Description: Halts the system.
# Description:       Halts the System.
# X-LFS-Provided-By: LFS
### END INIT INFO

case "${1}" in
    start)
        halt -d -f -i -p
        ;;

    *)
        echo "Usage: {start}"
        exit 1
        ;;
esac

# End halt
```

D.20. /etc/rc.d/init.d/template

```
#!/bin/sh
#####
# Begin scriptname
#
# Description :
#
# Authors      :
#
# Version      : LFS x.x
#
# Notes        :
#
#####
```

```

### BEGIN INIT INFO
# Provides:          template
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:
# Default-Stop:
# Short-Description:
# Description:
# X-LFS-Provided-By:
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Starting..."
        # if it is possible to use start_daemon
        start_daemon fully_qualified_path
        # if it is not possible to use start_daemon
        # (command to start the daemon is not simple enough)
        if ! pidofproc daemon_name_as_reported_by_ps >/dev/null; then
            command_to_start_the_service
        fi
        evaluate_retval
        ;;

    stop)
        log_info_msg "Stopping..."
        # if it is possible to use killproc
        killproc fully_qualified_path
        # if it is not possible to use killproc
        # (the daemon shouldn't be stopped by killing it)
        if pidofproc daemon_name_as_reported_by_ps >/dev/null; then
            command_to_stop_the_service
        fi
        evaluate_retval
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart}"
        exit 1
        ;;
esac

exit 0

# End scriptname

```

D.21. /etc/sysconfig/modules

```

#####
# Begin /etc/sysconfig/modules
#
# Description : Module auto-loading configuration
#
# Authors      :
#
# Version      : 00.00
#

```



```
# Notes      : The syntax of this file is as follows:
#             <module> [<arg1> <arg2> ...]
#
# Each module should be on its own line, and any options that you want
# passed to the module should follow it.  The line delimitator is either
# a space or a tab.
#####

# End /etc/sysconfig/modules
```

D.22. /etc/sysconfig/createfiles

```
#####
# Begin /etc/sysconfig/createfiles
#
# Description : Createfiles script config file
#
# Authors      :
#
# Version      : 00.00
#
# Notes        : The syntax of this file is as follows:
#                 if type is equal to "file" or "dir"
#                   <filename> <type> <permissions> <user> <group>
#                 if type is equal to "dev"
#                   <filename> <type> <permissions> <user> <group> <devtype>
#                   <major> <minor>
#
#                 <filename> is the name of the file which is to be created
#                 <type> is either file, dir, or dev.
#                   file creates a new file
#                   dir creates a new directory
#                   dev creates a new device
#                 <devtype> is either block, char or pipe
#                   block creates a block device
#                   char creates a character device
#                   pipe creates a pipe, this will ignore the <major> and
#                   <minor> fields
#                 <major> and <minor> are the major and minor numbers used for
#                 the device.
#####

# End /etc/sysconfig/createfiles
```

D.23. /etc/sysconfig/udev-retry

```
#####
# Begin /etc/sysconfig/udev_retry
#
# Description : udev_retry script configuration
#
# Authors      :
#
# Version      : 00.00
#
# Notes        : Each subsystem that may need to be re-triggered after mountfs
#                 runs should be listed in this file.  Probable subsystems to be
#                 listed here are rtc (due to /var/lib/hwclock/adjtime) and sound
#                 (due to both /var/lib/alsa/asound.state and /usr/sbin/alsactl).
#                 Entries are whitespace-separated.
#####

rtc

# End /etc/sysconfig/udev_retry
```

D.24. /sbin/ifup

```
#!/bin/sh
#####
# Begin /sbin/ifup
#
# Description : Interface Up
#
# Authors      : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpflaming@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
#
# Version      : LFS 7.7
#
# Notes        : The IFCONFIG variable is passed to the SERVICE script
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
#####

up()
{
    log_info_msg "Bringing up the ${1} interface..."

    if ip link show $1 > /dev/null 2>&1; then
        link_status=`ip link show $1`

        if [ -n "${link_status}" ]; then
            if ! echo "${link_status}" | grep -q UP; then
                ip link set $1 up
            fi
        fi

    else
        log_failure_msg "Interface ${IFACE} doesn't exist."
        exit 1
    fi

    evaluate_retval
}

RELEASE="7.7"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifup, version ${RELEASE}"

while [ $# -gt 0 ]; do
    case "$1" in
        --help | -h)      help="y"; break ;;

        --version | -V)  echo "${VERSTR}"; exit 0 ;;

        -*)               echo "ifup: ${1}: invalid option" >&2
                          echo "${USAGE}" >& 2
                          exit 2 ;;

        *)               break ;;
    esac
done

if [ -n "$help" ]; then
    echo "${VERSTR}"
    echo "${USAGE}"
    echo
    cat << HERE_EOF
ifup is used to bring up a network interface. The interface
```

```

parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

HERE_EOF
    exit 0
fi

file=/etc/sysconfig/ifconfig.${1}

# Skip backup files
[ "${file}" = "${file%*"~"}" ] || exit 0

. /lib/lsb/init-functions

if [ ! -r "${file}" ]; then
    log_failure_msg "Unable to bring up ${1} interface! ${file} is missing or cannot be accessed."
    exit 1
fi

. $file

if [ "$IFACE" = "" ]; then
    log_failure_msg "Unable to bring up ${1} interface! ${file} does not define an interface [IFACE]."
    exit 1
fi

# Do not process this service if started by boot, and ONBOOT
# is not set to yes
if [ "${IN_BOOT}" = "1" -a "${ONBOOT}" != "yes" ]; then
    exit 0
fi

# Bring up the interface
if [ "$VIRTINT" != "yes" ]; then
    up ${IFACE}
fi

for S in ${SERVICE}; do
    if [ ! -x "/lib/services/${S}" ]; then
        MSG="\nUnable to process ${file}. Either "
        MSG="${MSG}the SERVICE '${S}' was not present "
        MSG="${MSG}or cannot be executed."
        log_failure_msg "$MSG"
        exit 1
    fi
done

#if [ "${SERVICE}" = "wpa" ]; then log_success_msg; fi

# Create/configure the interface
for S in ${SERVICE}; do
    IFCONFIG=${file} /lib/services/${S} ${IFACE} up
done

# Set link up virtual interfaces
if [ "$VIRTINT" == "yes" ]; then
    up ${IFACE}
fi

# Bring up any additional interface components
for I in $INTERFACE_COMPONENTS; do up $I; done

# Set MTU if requested. Check if MTU has a "good" value.
if test -n "${MTU}"; then
    if [[ ${MTU} =~ ^[0-9]+$ ]] && [[ $MTU -ge 68 ]]; then
        for I in $IFACE $INTERFACE_COMPONENTS; do
            ip link set dev $I mtu $MTU;
        done
    fi
fi

```

```

done
else
    log_info_msg2 "Invalid MTU $MTU"
fi
fi

# Set the route default gateway if requested
if [ -n "${GATEWAY}" ]; then
    if ip route | grep -q default; then
        log_warning_msg "Gateway already setup; skipping."
    else
        log_info_msg "Adding default gateway ${GATEWAY} to the ${IFACE} interface..."
        ip route add default via ${GATEWAY} dev ${IFACE}
        evaluate_retval
    fi
fi

# End /sbin/ifup

```

D.25. /sbin/ifdown

```

#!/bin/bash
#####
# Begin /sbin/ifdown
#
# Description : Interface Down
#
# Authors      : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
# Notes        : the IFCONFIG variable is passed to the scripts found
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
#####

RELEASE="7.0"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifdown, version ${RELEASE}"

while [ $# -gt 0 ]; do
    case "$1" in
        --help | -h)      help="y"; break ;;

        --version | -V)   echo "${VERSTR}"; exit 0 ;;

        -*)               echo "ifup: ${1}: invalid option" >&2
                          echo "${USAGE}" >& 2
                          exit 2 ;;

        *)               break ;;
    esac
done

if [ -n "$help" ]; then
    echo "${VERSTR}"
    echo "${USAGE}"
    echo
    cat << HERE_EOF
ifdown is used to bring down a network interface. The interface
parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

```

```

HERE_EOF
    exit 0
fi

file=/etc/sysconfig/ifconfig.${1}

# Skip backup files
[ "${file}" = "${file%}"~" ] || exit 0

. /lib/lsb/init-functions

if [ ! -r "${file}" ]; then
    log_warning_msg "${file} is missing or cannot be accessed."
    exit 1
fi

. ${file}

if [ "$IFACE" = "" ]; then
    log_failure_msg "${file} does not define an interface [IFACE]."
    exit 1
fi

# We only need to first service to bring down the interface
S=`echo ${SERVICE} | cut -f1 -d" "`

if ip link show ${IFACE} > /dev/null 2>&1; then
    if [ -n "${S}" -a -x "/lib/services/${S}" ]; then
        IFCONFIG=${file} /lib/services/${S} ${IFACE} down
    else
        MSG="Unable to process ${file}. Either "
        MSG="${MSG}the SERVICE variable was not set "
        MSG="${MSG}or the specified service cannot be executed."
        log_failure_msg "$MSG"
        exit 1
    fi
else
    log_warning_msg "Interface ${1} doesn't exist."
fi

# Leave the interface up if there are additional interfaces in the device
link_status=`ip link show ${IFACE} 2>/dev/null`

if [ -n "${link_status}" ]; then
    if [ "$(echo "${link_status}" | grep UP)" != "" ]; then
        if [ "$(ip addr show ${IFACE} | grep 'inet ')" == "" ]; then
            log_info_msg "Bringing down the ${IFACE} interface..."
            ip link set ${IFACE} down
            evaluate_retval
        fi
    fi
fi

# End /sbin/ifdown

```

D.26. /lib/services/ipv4-static

```

#!/bin/sh
#####
# Begin /lib/services/ipv4-static
#
# Description : IPV4 Static Boot Script
#
# Authors      : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kp Fleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#

```

```

# Version      : LFS 7.0
#
#####

. /lib/lsb/init-functions
. ${IFCONFIG}

if [ -z "${IP}" ]; then
    log_failure_msg "\nIP variable missing from ${IFCONFIG}, cannot continue."
    exit 1
fi

if [ -z "${PREFIX}" -a -z "${PEER}" ]; then
    log_warning_msg "\nPREFIX variable missing from ${IFCONFIG}, assuming 24."
    PREFIX=24
    args="${args} ${IP}/${PREFIX}"
elif [ -n "${PREFIX}" -a -n "${PEER}" ]; then
    log_failure_msg "\nPREFIX and PEER both specified in ${IFCONFIG}, cannot continue."
    exit 1
elif [ -n "${PREFIX}" ]; then
    args="${args} ${IP}/${PREFIX}"
elif [ -n "${PEER}" ]; then
    args="${args} ${IP} peer ${PEER}"
fi

if [ -n "${LABEL}" ]; then
    args="${args} label ${LABEL}"
fi

if [ -n "${BROADCAST}" ]; then
    args="${args} broadcast ${BROADCAST}"
fi

case "${2}" in
    up)
        if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" = "" ]; then
            log_info_msg "Adding IPv4 address ${IP} to the ${1} interface..."
            ip addr add ${args} dev ${1}
            evaluate_retval
        else
            log_warning_msg "Cannot add IPv4 address ${IP} to ${1}.  Already present."
        fi
        ;;

    down)
        if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" != "" ]; then
            log_info_msg "Removing IPv4 address ${IP} from the ${1} interface..."
            ip addr del ${args} dev ${1}
            evaluate_retval
        fi

        if [ -n "${GATEWAY}" ]; then
            # Only remove the gateway if there are no remaining ipv4 addresses
            if [ "$(ip addr show ${1} 2>/dev/null | grep 'inet ')" != "" ]; then
                log_info_msg "Removing default gateway..."
                ip route del default
                evaluate_retval
            fi
        fi
        ;;

    *)
        echo "Usage: ${0} [interface] {up|down}"
        exit 1

```

```
;;
esac

# End /lib/services/ipv4-static
```

D.27. /lib/services/ipv4-static-route

```
#!/bin/sh
#####
# Begin /lib/services/ipv4-static-route
#
# Description : IPV4 Static Route Script
#
# Authors      : Kevin P. Fleming - kpflaming@linuxfromscratch.org
#              : DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

. /lib/lsb/init-functions
. ${IFCONFIG}

case "${TYPE}" in
    (" | "network")
        need_ip=1
        need_gateway=1
        ;;

    ("default")
        need_gateway=1
        args="${args} default"
        desc="default"
        ;;

    ("host")
        need_ip=1
        ;;

    ("unreachable")
        need_ip=1
        args="${args} unreachable"
        desc="unreachable "
        ;;

    (*)
        log_failure_msg "Unknown route type (${TYPE}) in ${IFCONFIG}, cannot continue."
        exit 1
        ;;
esac

if [ -n "${GATEWAY}" ]; then
    MSG="The GATEWAY variable cannot be set in ${IFCONFIG} for static routes.\n"
    log_failure_msg "$MSG Use STATIC_GATEWAY only, cannot continue"
    exit 1
fi

if [ -n "${need_ip}" ]; then
    if [ -z "${IP}" ]; then
        log_failure_msg "IP variable missing from ${IFCONFIG}, cannot continue."
        exit 1
    fi

    if [ -z "${PREFIX}" ]; then
        log_failure_msg "PREFIX variable missing from ${IFCONFIG}, cannot continue."
        exit 1
    fi
fi
```

```

fi

args="${args} ${IP}/${PREFIX}"
desc="${desc}${IP}/${PREFIX}"
fi

if [ -n "${need_gateway}" ]; then
  if [ -z "${STATIC_GATEWAY}" ]; then
    log_failure_msg "STATIC_GATEWAY variable missing from ${IFCONFIG}, cannot continue."
    exit 1
  fi
  args="${args} via ${STATIC_GATEWAY}"
fi

if [ -n "${SOURCE}" ]; then
  args="${args} src ${SOURCE}"
fi

case "${2}" in
  up)
    log_info_msg "Adding '${desc}' route to the ${1} interface..."
    ip route add ${args} dev ${1}
    evaluate_retval
    ;;
  down)
    log_info_msg "Removing '${desc}' route from the ${1} interface..."
    ip route del ${args} dev ${1}
    evaluate_retval
    ;;
  *)
    echo "Usage: ${0} [interface] {up|down}"
    exit 1
    ;;
esac

# End /lib/services/ipv4-static-route

```


Tillegg E. Udev konfigurasjonsregler

Reglene i dette vedlegget er listet opp for enkelhets skyld. Installasjon gjøres normalt via instruksjoner i Seksjon 8.76, «Udev fra Systemd-257.3»

E.1. 55-lfs.rules

```
# /etc/udev/rules.d/55-lfs.rules: Rule definitions for LFS.

# Core kernel devices

# This causes the system clock to be set as soon as /dev/rtc becomes available.
SUBSYSTEM=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"
KERNEL=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"
```

Tillegg F. LFS lisenser

Denne boken er lisensiert under Creative Commons Attribution-NonCommercial-ShareAlike 2.0 Lisensen.

Datainstruksjoner kan trekkes ut fra boken under MIT Lisensen.

F.1. Creative Commons License

Creative Commons Legal Code

Attribution-NonCommercial-ShareAlike 2.0



Viktig

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.
- 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, MERCHANTABILITY, 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.

**Viktig**

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/>.

F.2. The MIT License

Copyright © 1999-2025 Gerard Beekmans

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Register

Pakker

- Acl: 134
- Attr: 133
- Autoconf: 171
- Automake: 172
- Bash: 157
 - tools: 61
- Bash: 157
 - tools: 61
- Bc: 118
- Binutils: 126
 - tools, pass 1: 47
 - tools, pass 2: 74
- Binutils: 126
 - tools, pass 1: 47
 - tools, pass 2: 74
- Binutils: 126
 - tools, pass 1: 47
 - tools, pass 2: 74
- Bison: 155
 - tools: 84
- Bison: 155
 - tools: 84
- Bootscripts: 237
 - usage: 247
- Bootscripts: 237
 - usage: 247
- Bzip2: 108
- Check: 190
- Coreutils: 185
 - tools: 62
- Coreutils: 185
 - tools: 62
- DejaGNU: 124
- Diffutils: 191
 - tools: 63
- Diffutils: 191
 - tools: 63
- E2fsprogs: 228
- Expat: 162
- Expect: 122
- File: 114
 - tools: 64
- File: 114
 - tools: 64
- Findutils: 193
 - tools: 65
- Findutils: 193
 - tools: 65
- Flex: 119
- Flit-core: 179
- Gawk: 192
 - tools: 66
- Gawk: 192
 - tools: 66
- GCC: 142
 - tools, libstdc++ pass 1: 56
 - tools, pass 1: 49
 - tools, pass 2: 75
- GCC: 142
 - tools, libstdc++ pass 1: 56
 - tools, pass 1: 49
 - tools, pass 2: 75
- GCC: 142
 - tools, libstdc++ pass 1: 56
 - tools, pass 1: 49
 - tools, pass 2: 75
- GCC: 142
 - tools, libstdc++ pass 1: 56
 - tools, pass 1: 49
 - tools, pass 2: 75
- GDBM: 160
- Gettext: 153
 - tools: 83
- Gettext: 153
 - tools: 83
- Glibc: 99
 - tools: 53
- Glibc: 99
 - tools: 53
- GMP: 129
- Gperf: 161
- Grep: 156
 - tools: 67
- Grep: 156
 - tools: 67
- Groff: 194
- GRUB: 197
- Gzip: 199
 - tools: 68
- Gzip: 199
 - tools: 68
- Iana-Etc: 98
- Inetutils: 163
- Intltool: 170
- IPRoute2: 200
- Jinja2: 214

Kbd: 202
 Kmod: 184
 Less: 165
 Libcap: 135
 Libelf: 175
 libffi: 176
 Libpipeline: 204
 Libtool: 159
 Libxcrypt: 136
 Linux: 261
 tools, API headers: 52
 Linux: 261
 tools, API headers: 52
 Lz4: 112
 M4: 117
 tools: 58
 M4: 117
 tools: 58
 Make: 205
 tools: 69
 Make: 205
 tools: 69
 Man-DB: 218
 Man-pages: 97
 MarkupSafe: 213
 Meson: 183
 MPC: 132
 MPFR: 131
 Ncurses: 148
 tools: 59
 Ncurses: 148
 tools: 59
 Ninja: 182
 OpenSSL: 173
 Patch: 206
 tools: 70
 Patch: 206
 tools: 70
 Perl: 166
 tools: 85
 Perl: 166
 tools: 85
 Pkgconf: 125
 Procps-ng: 221
 Psmisc: 152
 Python: 177
 temporary: 86
 Python: 177
 temporary: 86
 rc.site: 253
 Readline: 115
 Sed: 151
 tools: 71
 Sed: 151
 tools: 71
 Setuptools: 181
 Shadow: 138
 configuring: 139
 Shadow: 138
 configuring: 139
 Syslogd: 231
 configuring: 231
 Syslogd: 231
 configuring: 231
 SysVinit: 232
 configuring: 248
 SysVinit: 232
 configuring: 248
 Tar: 207
 tools: 72
 Tar: 207
 tools: 72
 Tcl: 120
 Texinfo: 208
 temporary: 87
 Texinfo: 208
 temporary: 87
 Udev: 215
 configuring: 216
 usage: 239
 Udev: 215
 configuring: 216
 usage: 239
 Udev: 215
 configuring: 216
 usage: 239
 Util-linux: 223
 tools: 88
 Util-linux: 223
 tools: 88
 Vim: 210
 wheel: 180
 XML::Parser: 169
 Xz: 110
 tools: 73
 Xz: 110
 tools: 73
 Zlib: 107
 zstd: 113

Programmer

[: 185, 186
 2to3: 177
 accessdb: 218, 219
 aclocal: 172, 172
 aclocal-1.17: 172, 172
 addftinfo: 194, 194
 addpart: 223, 224
 addr2line: 126, 127
 afmtodit: 194, 194
 agetty: 223, 224
 apropos: 218, 219
 ar: 126, 127
 as: 126, 127
 attr: 133, 133
 autoconf: 171, 171
 autoheader: 171, 171
 autom4te: 171, 171
 automake: 172, 172
 automake-1.17: 172, 172
 autopoint: 153, 153
 autoreconf: 171, 171
 autoscan: 171, 171
 autoupdate: 171, 171
 awk: 192, 192
 b2sum: 185, 186
 badblocks: 228, 229
 base64: 185, 186, 185, 186
 base64: 185, 186, 185, 186
 basename: 185, 186
 basenc: 185, 186
 bash: 157, 158
 bashbug: 157, 158
 bc: 118, 118
 bison: 155, 155
 blkdiscard: 223, 224
 blkid: 223, 224
 blkzone: 223, 224
 blockdev: 223, 224
 bomtool: 125, 125
 bootlogd: 232, 232
 bridge: 200, 200
 bunzip2: 108, 109
 bzcac: 108, 109
 bzcac: 108, 109
 bzcmp: 108, 109
 bzdiff: 108, 109
 bzegrep: 108, 109
 bzfgrep: 108, 109
 bzgrep: 108, 109
 bzip2: 108, 109
 bzip2recover: 108, 109
 bzless: 108, 109
 bzmorc: 108, 109
 c++: 142, 146
 c++filt: 126, 127
 cal: 223, 224
 capsh: 135, 135
 captinfo: 148, 149
 cat: 185, 186
 catman: 218, 219
 cc: 142, 146
 cfdisk: 223, 224
 chacl: 134, 134
 chage: 138, 140
 chatr: 228, 229
 chcon: 185, 186
 chcpu: 223, 224
 checkmk: 190, 190
 chem: 194, 194
 chfn: 138, 140
 chpasswd: 138, 140
 chgrp: 185, 186
 chmem: 223, 224
 chmod: 185, 186
 choom: 223, 224
 chown: 185, 186
 chpasswd: 138, 140
 chroot: 185, 186
 chrt: 223, 224
 chsh: 138, 140
 chvt: 202, 203
 cksum: 185, 186
 clear: 148, 149
 cmp: 191, 191
 col: 223, 224
 colcrt: 223, 224
 colrm: 223, 224
 column: 223, 224
 comm: 185, 186
 compile_et: 228, 229
 corelist: 166, 167
 cp: 185, 187
 cpan: 166, 167
 cpp: 142, 146
 csplit: 185, 187
 ctrlaltdel: 223, 224
 ctstat: 200, 200
 cut: 185, 187
 c_rehash: 173, 174
 date: 185, 187

dc: 118, 118
 dd: 185, 187
 deallocvt: 202, 203
 debugfs: 228, 229
 dejagnu: 124, 124
 delpart: 223, 224
 depmod: 184, 184
 df: 185, 187
 diff: 191, 191
 diff3: 191, 191
 dir: 185, 187
 dircolors: 185, 187
 dirname: 185, 187
 dmesg: 223, 224
 dnsdomainname: 163, 164
 du: 185, 187
 dumpe2fs: 228, 229
 dumpkeys: 202, 203
 e2freefrag: 228, 229
 e2fsck: 228, 229
 e2image: 228, 229
 e2label: 228, 229
 e2mmpstatus: 228, 229
 e2scrub: 228, 229
 e2scrub_all: 228, 229
 e2undo: 228, 229
 e4crypt: 228, 229
 e4defrag: 228, 229
 echo: 185, 187
 egrep: 156, 156
 eject: 223, 225
 elfedit: 126, 127
 enc2xs: 166, 167
 encguess: 166, 167
 env: 185, 187
 envsubst: 153, 153
 eqn: 194, 194
 eqn2graph: 194, 194
 ex: 210, 211
 expand: 185, 187
 expect: 122, 123
 expiry: 138, 140
 expr: 185, 187
 factor: 185, 187
 faillog: 138, 140
 fallocate: 223, 225
 false: 185, 187
 fdisk: 223, 225
 fgconsole: 202, 203
 fgrep: 156, 156
 file: 114, 114
 filefrag: 228, 229
 findcore: 223, 225
 find: 193, 193
 findfs: 223, 225
 findmnt: 223, 225
 flex: 119, 119
 flex++: 119, 119
 flock: 223, 225
 fmt: 185, 187
 fold: 185, 187
 free: 221, 221
 fsck: 223, 225
 fsck.cramfs: 223, 225
 fsck.ext2: 228, 229
 fsck.ext3: 228, 229
 fsck.ext4: 228, 229
 fsck.minix: 223, 225
 fsfreeze: 223, 225
 fstab-decode: 232, 232
 fstrim: 223, 225
 ftp: 163, 164
 fuser: 152, 152
 g++: 142, 146
 gawk: 192, 192
 gawk-5.3.1: 192, 192
 gcc: 142, 146
 gc-ar: 142, 146
 gc-nm: 142, 146
 gc-ranlib: 142, 146
 gcov: 142, 146
 gcov-dump: 142, 146
 gcov-tool: 142, 146
 gdbmtool: 160, 160
 gdbm_dump: 160, 160
 gdbm_load: 160, 160
 gdiffmk: 194, 194
 gencat: 99, 105
 genl: 200, 200
 getcap: 135, 135
 getconf: 99, 105
 getent: 99, 105
 getfacl: 134, 134
 getfattr: 133, 133
 getkeycodes: 202, 203
 getopt: 223, 225
 getpcaps: 135, 135
 getsubids: 138, 140
 gettext: 153, 153
 gettext.sh: 153, 153

gettextize: 153, 153
 glilypond: 194, 194
 gpasswd: 138, 140
 gperf: 161, 161
 gperl: 194, 194
 gpinyin: 194, 194
 gprof: 126, 127
 gprofng: 126, 127
 grap2graph: 194, 194
 grep: 156, 156
 grn: 194, 195
 grodvi: 194, 195
 groff: 194, 195
 groffer: 194, 195
 grog: 194, 195
 grolbp: 194, 195
 grolj4: 194, 195
 gropdf: 194, 195
 grops: 194, 195
 grotty: 194, 195
 groupadd: 138, 140
 groupdel: 138, 140
 groupmems: 138, 140
 groupmod: 138, 140
 groups: 185, 187
 grpck: 138, 140
 grpconv: 138, 140
 grpunconv: 138, 140
 grub-bios-setup: 197, 198
 grub-editenv: 197, 198
 grub-file: 197, 198
 grub-fstest: 197, 198
 grub-glue-efi: 197, 198
 grub-install: 197, 198
 grub-kbdcomp: 197, 198
 grub-macbless: 197, 198
 grub-menulst2cfg: 197, 198
 grub-mkconfig: 197, 198
 grub-mkimage: 197, 198
 grub-mklayout: 197, 198
 grub-mknetdir: 197, 198
 grub-mkpasswd-pbkdf2: 197, 198
 grub-mkrelpath: 197, 198
 grub-mkrescue: 197, 198
 grub-mkstandalone: 197, 198
 grub-ofpathname: 197, 198
 grub-probe: 197, 198
 grub-reboot: 197, 198
 grub-render-label: 197, 198
 grub-script-check: 197, 198
 grub-set-default: 197, 198
 grub-setup: 197, 198
 grub-syslinux2cfg: 197, 198
 gunzip: 199, 199
 gzexe: 199, 199
 gzip: 199, 199
 h2ph: 166, 167
 h2xs: 166, 167
 halt: 232, 232
 hardlink: 223, 225
 head: 185, 187
 hexdump: 223, 225
 hostid: 185, 187
 hostname: 163, 164
 hpftodit: 194, 195
 hwclock: 223, 225
 i386: 223, 225
 iconv: 99, 105
 iconvconfig: 99, 105
 id: 185, 187
 idle3: 177
 ifconfig: 163, 164
 ifnames: 171, 171
 ifstat: 200, 200
 indxbib: 194, 195
 info: 208, 208
 infocmp: 148, 149
 infotocap: 148, 149
 init: 232, 232
 insmod: 184, 184
 install: 185, 187
 install-info: 208, 208
 instmodsh: 166, 167
 intltool-extract: 170, 170
 intltool-merge: 170, 170
 intltool-prepare: 170, 170
 intltool-update: 170, 170
 intltoolize: 170, 170
 ionice: 223, 225
 ip: 200, 200
 ipcmk: 223, 225
 ipcrm: 223, 225
 ipcs: 223, 225
 irqtop: 223, 225
 isosize: 223, 225
 join: 185, 187
 json_pp: 166, 167
 kbdfinfo: 202, 203
 kbdrate: 202, 203
 kbd_mode: 202, 203

kill: 223, 225
 killall: 152, 152
 killall5: 232, 232
 kmod: 184, 184
 last: 223, 225
 lastb: 223, 225
 ld: 126, 127
 ld.bfd: 126, 127
 ldattach: 223, 225
 ldconfig: 99, 105
 ldd: 99, 105
 lddlibc4: 99, 105
 less: 165, 165
 lessecho: 165, 165
 lesskey: 165, 165
 lex: 119, 119
 lexgrog: 218, 219
 lfskernel-6.13.4: 261, 266
 libasan: 142, 146
 libatomic: 142, 146
 libcc1: 142, 146
 libnetcfg: 166, 167
 libtool: 159, 159
 libtoolize: 159, 159
 link: 185, 187
 linux32: 223, 225
 linux64: 223, 225
 lkbib: 194, 195
 ln: 185, 187
 lnstat: 200, 200
 loadkeys: 202, 203
 loadunimap: 202, 203
 locale: 99, 105
 localedef: 99, 105
 locate: 193, 193
 logger: 223, 225
 login: 138, 140
 logname: 185, 187
 logoutd: 138, 140
 logsave: 228, 229
 look: 223, 225
 lookbib: 194, 195
 losetup: 223, 225
 ls: 185, 187
 lsattr: 228, 230
 lsblk: 223, 225
 lspcu: 223, 225
 lsfd: 223, 225
 lsipc: 223, 225
 lsirq: 223, 226
 lslocks: 223, 226
 lslogins: 223, 226
 lsmem: 223, 226
 lsmod: 184, 184
 lsns: 223, 226
 lto-dump: 142, 146
 lz4: 112, 112
 lz4c: 112, 112
 lz4cat: 112, 112
 lzcat: 110, 110
 lzcmp: 110, 110
 lzdiff: 110, 110
 lzegrep: 110, 110
 lzfgrep: 110, 110
 lzgrep: 110, 110
 lzless: 110, 110
 lzma: 110, 110
 lzmadec: 110, 110
 lzmainfo: 110, 110
 lzmore: 110, 110
 m4: 117, 117
 make: 205, 205
 makedb: 99, 105
 makeinfo: 208, 208
 man: 218, 219
 man-recode: 218, 219
 mandb: 218, 220
 manpath: 218, 220
 mapscrn: 202, 203
 mcookie: 223, 226
 md5sum: 185, 187
 msg: 223, 226
 meson: 183, 183
 mkdir: 185, 187
 mke2fs: 228, 230
 mkfifo: 185, 187
 mkfs: 223, 226
 mkfs.bfs: 223, 226
 mkfs.cramfs: 223, 226
 mkfs.ext2: 228, 230
 mkfs.ext3: 228, 230
 mkfs.ext4: 228, 230
 mkfs.minix: 223, 226
 mklost+found: 228, 230
 mknod: 185, 187
 mkswap: 223, 226
 mktemp: 185, 187
 mk_cmds: 228, 230
 mmroff: 194, 195
 modinfo: 184, 184

modprobe: 184, 184
 more: 223, 226
 mount: 223, 226
 mountpoint: 223, 226
 msgattrib: 153, 153
 msgcat: 153, 153
 msgcmp: 153, 153
 msgcomm: 153, 153
 msgconv: 153, 153
 msgen: 153, 154
 msgexec: 153, 154
 msgfilter: 153, 154
 msgfmt: 153, 154
 msggrep: 153, 154
 msginit: 153, 154
 msgmerge: 153, 154
 msgunfmt: 153, 154
 msguniq: 153, 154
 mtrace: 99, 105
 mv: 185, 188
 namei: 223, 226
 ncursesw6-config: 148, 149
 neqn: 194, 195
 newgidmap: 138, 140
 newgrp: 138, 140
 newuidmap: 138, 140
 newusers: 138, 141
 ngettext: 153, 154
 nice: 185, 188
 ninja: 182, 182
 nl: 185, 188
 nm: 126, 127
 nohup: 185, 188
 nologin: 138, 141
 nproc: 185, 188
 nroff: 194, 195
 nsenter: 223, 226
 nstat: 200, 201
 numfmt: 185, 188
 objcopy: 126, 127
 objdump: 126, 127
 od: 185, 188
 openssl: 173, 174
 openvt: 202, 203
 partx: 223, 226
 passwd: 138, 141
 paste: 185, 188
 patch: 206, 206
 pathchk: 185, 188
 pcprofiledump: 99, 105
 pdfmom: 194, 195
 pdfroff: 194, 195
 pdftexi2dvi: 208, 209
 peekfd: 152, 152
 perl: 166, 167
 perl5.40.1: 166, 167
 perlbug: 166, 167
 perldoc: 166, 167
 perlivp: 166, 167
 perlthanks: 166, 167
 pfbtops: 194, 195
 pgrep: 221, 221
 pic: 194, 195
 pic2graph: 194, 195
 picconv: 166, 167
 pidof: 221, 221
 ping: 163, 164
 ping6: 163, 164
 pinky: 185, 188
 pip3: 177
 pivot_root: 223, 226
 pkgconf: 125, 125
 pkill: 221, 221
 pl2pm: 166, 167
 pldd: 99, 105
 pmap: 221, 221
 pod2html: 166, 167
 pod2man: 166, 167
 pod2texi: 208, 209
 pod2text: 166, 167
 pod2usage: 166, 167
 podchecker: 166, 167
 podselect: 166, 167
 post-grohtml: 194, 195
 poweroff: 232, 232
 pr: 185, 188
 pre-grohtml: 194, 195
 preconv: 194, 195
 printenv: 185, 188
 printf: 185, 188
 prlimit: 223, 226
 prove: 166, 167
 prtstat: 152, 152
 ps: 221, 221
 psfaddtable: 202, 203
 psfgettable: 202, 203
 psfstriptime: 202, 203
 psfxtable: 202, 203
 pslog: 152, 152
 pstree: 152, 152

pstree.x11: 152, 152
 ptar: 166, 167
 ptardiff: 166, 167
 ptargrep: 166, 167
 ptx: 185, 188
 pwck: 138, 141
 pwconv: 138, 141
 pwd: 185, 188
 pwdx: 221, 221
 pwunconv: 138, 141
 pydoc3: 177
 python3: 177
 ranlib: 126, 127
 readelf: 126, 127
 readlink: 185, 188
 readprofile: 223, 226
 realpath: 185, 188
 reboot: 232, 232
 recode-sr-latin: 153, 154
 refer: 194, 195
 rename: 223, 226
 renice: 223, 226
 reset: 148, 149
 resize2fs: 228, 230
 resizepart: 223, 226
 rev: 223, 226
 rfcill: 223, 226
 rm: 185, 188
 rmdir: 185, 188
 rmmod: 184, 184
 roff2dvi: 194, 195
 roff2html: 194, 195
 roff2pdf: 194, 195
 roff2ps: 194, 196
 roff2text: 194, 196
 roff2x: 194, 196
 routel: 200, 201
 rtacct: 200, 201
 rtcwake: 223, 226
 rtmon: 200, 201
 rtpr: 200, 201
 rtstat: 200, 201
 runcon: 185, 188
 runlevel: 232, 232
 runttest: 124, 124
 rview: 210, 212
 rvim: 210, 212
 script: 223, 226
 scriptlive: 223, 226
 scriptreplay: 223, 226
 sdiff: 191, 191
 sed: 151, 151
 seq: 185, 188
 setarch: 223, 226
 setcap: 135, 135
 setfacl: 134, 134
 setfattr: 133, 133
 setfont: 202, 203
 setkeycodes: 202, 203
 setleds: 202, 203
 setmetamode: 202, 203
 setsid: 223, 226
 setterm: 223, 226
 setvtrgb: 202, 203
 sfdisk: 223, 226
 sg: 138, 141
 sh: 157, 158
 sha1sum: 185, 188
 sha224sum: 185, 188
 sha256sum: 185, 188
 sha384sum: 185, 188
 sha512sum: 185, 188
 shasum: 166, 167
 showconsolefont: 202, 203
 showkey: 202, 203
 shred: 185, 188
 shuf: 185, 188
 shutdown: 232, 232
 size: 126, 127
 slabtop: 221, 221
 sleep: 185, 188
 sln: 99, 105
 soelim: 194, 196
 sort: 185, 188
 sotruss: 99, 105
 splain: 166, 167
 split: 185, 188
 sprof: 99, 105
 ss: 200, 201
 stat: 185, 188
 stdbuf: 185, 188
 strings: 126, 127
 strip: 126, 127
 stty: 185, 188
 su: 138, 141
 sulogin: 223, 226
 sum: 185, 188
 swapon: 223, 226
 swaponoff: 223, 226
 swapon: 223, 226

switch_root: 223, 227
 sync: 185, 188
 sysctl: 221, 222
 syslogd: 231, 231
 tabs: 148, 149
 tac: 185, 188
 tail: 185, 189
 talk: 163, 164
 tar: 207, 207
 taskset: 223, 227
 tbl: 194, 196
 tc: 200, 201
 tclsh: 120, 121
 tclsh8.6: 120, 121
 tee: 185, 189
 telinit: 232, 232
 telnet: 163, 164
 test: 185, 189
 texi2dvi: 208, 209
 texi2pdf: 208, 209
 texi2any: 208, 209
 texindex: 208, 209
 tfmtodit: 194, 196
 tftp: 163, 164
 tic: 148, 149
 timeout: 185, 189
 tload: 221, 222
 toe: 148, 149
 top: 221, 222
 touch: 185, 189
 tput: 148, 149
 tr: 185, 189
 traceroute: 163, 164
 troff: 194, 196
 true: 185, 189
 truncate: 185, 189
 tset: 148, 149
 tsort: 185, 189
 tty: 185, 189
 tune2fs: 228, 230
 tzselect: 99, 105
 uclampset: 223, 227
 udev-hwdb: 215, 217
 udevadm: 215, 217
 udevd: 215, 217
 ul: 223, 227
 umount: 223, 227
 uname: 185, 189
 uname26: 223, 227
 uncompress: 199, 199
 unexpand: 185, 189
 unicode_start: 202, 203
 unicode_stop: 202, 203
 uniq: 185, 189
 unlink: 185, 189
 unlz4: 112, 112
 unlzma: 110, 110
 unshare: 223, 227
 unxz: 110, 111
 updatedb: 193, 193
 uptime: 221, 222
 useradd: 138, 141
 userdel: 138, 141
 usermod: 138, 141
 users: 185, 189
 utmpdump: 223, 227
 uuid: 223, 227
 uuidgen: 223, 227
 uuidparse: 223, 227
 vdir: 185, 189
 vi: 210, 212
 view: 210, 212
 vigr: 138, 141
 vim: 210, 212
 vimdiff: 210, 212
 vimtutor: 210, 212
 vipw: 138, 141
 vmstat: 221, 222
 w: 221, 222
 wall: 223, 227
 watch: 221, 222
 wc: 185, 189
 wdctl: 223, 227
 whatis: 218, 220
 wheel: 180
 whereis: 223, 227
 who: 185, 189
 whoami: 185, 189
 wipefs: 223, 227
 x86_64: 223, 227
 xargs: 193, 193
 xgettext: 153, 154
 xmlwf: 162, 162
 xsubpp: 166, 168
 xtrace: 99, 105
 xxd: 210, 212
 xz: 110, 111
 xzcat: 110, 111
 xzcmp: 110, 111
 xzdec: 110, 111

xzdiff: 110, 111
 xzegrep: 110, 111
 xzfgrep: 110, 111
 xzgrep: 110, 111
 xzless: 110, 111
 xzmore: 110, 111
 yacc: 155, 155
 yes: 185, 189
 zcat: 199, 199
 zcmp: 199, 199
 zdiff: 199, 199
 zdump: 99, 105
 zegrep: 199, 199
 zfgrep: 199, 199
 zforce: 199, 199
 zgrep: 199, 199
 zic: 99, 105
 zipdetails: 166, 168
 zless: 199, 199
 zmore: 199, 199
 znew: 199, 199
 zramctl: 223, 227
 zstd: 113, 113
 zstdgrep: 113, 113
 zstdless: 113, 113

Biblioteker

Expat: 169, 169
 ld-2.41.so: 99, 105
 libacl: 134, 134
 libanl: 99, 105
 libasprintf: 153, 154
 libattr: 133, 133
 libbfd: 126, 127
 libblkid: 223, 227
 libBrokenLocale: 99, 105
 libbz2: 108, 109
 libc: 99, 105
 libcap: 135, 135
 libcheck: 190, 190
 libcom_err: 228, 230
 libcrypt: 136, 137
 libcrypto.so: 173, 174
 libctf: 126, 127
 libctf-nobfd: 126, 128
 libc_malloc_debug: 99, 105
 libdl: 99, 105
 libe2p: 228, 230
 libelf: 175, 175
 libexpat: 162, 162

libexpect-5.45.4: 122, 123
 libext2fs: 228, 230
 libfdisk: 223, 227
 libffi: 176
 libfl: 119, 119
 libformw: 148, 150
 libg: 99, 105
 libgcc: 142, 146
 libgcov: 142, 146
 libgdbm: 160, 160
 libgdbm_compat: 160, 160
 libgettextlib: 153, 154
 libgettextpo: 153, 154
 libgettextsrc: 153, 154
 libgmp: 129, 130
 libgmpxx: 129, 130
 libgomp: 142, 146
 libgprofng: 126, 128
 libhistory: 115, 115
 libhwasan: 142, 146
 libitm: 142, 146
 libkmod: 184
 liblsan: 142, 146
 libltdl: 159, 159
 liblto_plugin: 142, 146
 liblz4: 112, 112
 liblzma: 110, 111
 libm: 99, 105
 libmagic: 114, 114
 libman: 218, 220
 libmandb: 218, 220
 libmcheck: 99, 105
 libmemusage: 99, 105
 libmenuw: 148, 150
 libmount: 223, 227
 libmpc: 132, 132
 libmpfr: 131, 131
 libmvec: 99, 105
 libncurses++w: 148, 150
 libncursesw: 148, 149
 libnsl: 99, 106
 libnss_*: 99, 106
 libopcodes: 126, 128
 libpanelw: 148, 150
 libpcprofile: 99, 106
 libpipeline: 204
 libpkgconf: 125, 125
 libproc-2: 221, 222
 libpsx: 135, 135
 libpthread: 99, 106

libquadmath: 142, 146
 libreadline: 115, 116
 libresolv: 99, 106
 librt: 99, 106
 libsframe: 126, 128
 libsmartcols: 223, 227
 libss: 228, 230
 libssl.so: 173, 174
 libssp: 142, 146
 libstdbuf: 185, 189
 libstdc++: 142, 146
 libstdc++exp: 142, 146
 libstdc++fs: 142, 146
 libsubid: 138, 141
 libsupc++: 142, 146
 libtcl8.6.so: 120, 121
 libtclstub8.6.a: 120, 121
 libtextstyle: 153, 154
 libthread_db: 99, 106
 libtsan: 142, 146
 libubsan: 142, 147
 libudev: 215, 217
 libutil: 99, 106
 libuuid: 223, 227
 liby: 155, 155
 libz: 107, 107
 libzstd: 113, 113
 preloadable_libintl: 153, 154

Skript

checkfs: 237, 237
 cleanfs: 237, 237
 console: 237, 237
 configuring: 250
 console: 237, 237
 configuring: 250
 File creation at boot
 configuring: 252
 functions: 237, 237
 halt: 237, 237
 hostname
 configuring: 246
 ifdown: 237, 237
 ifup: 237, 237
 ipv4-static: 237, 237
 localnet: 237, 237
 /etc/hosts: 246
 localnet: 237, 237
 /etc/hosts: 246
 modules: 237, 237

mountfs: 237, 237
 mountvirtfs: 237, 237
 network: 237, 237
 /etc/hosts: 246
 configuring: 244
 network: 237, 237
 /etc/hosts: 246
 configuring: 244
 network: 237, 237
 /etc/hosts: 246
 configuring: 244
 rc: 237, 237
 reboot: 237, 237
 sendsignals: 237, 237
 setclock: 237, 237
 configuring: 249
 setclock: 237, 237
 configuring: 249
 swap: 237, 238
 sysctl: 237, 238
 sysklogd: 237, 238
 configuring: 253
 sysklogd: 237, 238
 configuring: 253
 template: 237, 238
 udev: 237, 238
 udev_retry: 237, 238
 dwp: 126, 127

Andre

/boot/config-6.13.4: 261, 266
 /boot/System.map-6.13.4: 261, 266
 /dev/*: 77
 /etc/fstab: 259
 /etc/group: 80
 /etc/hosts: 246
 /etc/inittab: 248
 /etc/inputrc: 256
 /etc/ld.so.conf: 104
 /etc/lfs-release: 270
 /etc/localtime: 103
 /etc/lsb-release: 270
 /etc/mke2fs.conf: 229
 /etc/modprobe.d/usb.conf: 266
 /etc/nsswitch.conf: 103
 /etc/os-release: 270
 /etc/passwd: 80
 /etc/profile: 255
 /etc/protocols: 98
 /etc/resolv.conf: 245

/etc/services: 98
/etc/syslog.conf: 231
/etc/udev: 215, 217
/etc/udev/hwdb.bin: 216
/etc/vimrc: 211
/run/utmp: 80
/usr/include/asm-generic/*.h: 52, 52
/usr/include/asm/*.h: 52, 52
/usr/include/drm/*.h: 52, 52
/usr/include/linux/*.h: 52, 52
/usr/include/misc/*.h: 52, 52
/usr/include/mtd/*.h: 52, 52
/usr/include/rdma/*.h: 52, 52
/usr/include/scsi/*.h: 52, 52
/usr/include/sound/*.h: 52, 52
/usr/include/video/*.h: 52, 52
/usr/include/xen/*.h: 52, 52
/var/log/btmp: 80
/var/log/lastlog: 80
/var/log/wtmp: 80
/etc/shells: 257
man pages: 97, 97