Linux From Scratch

Versjon 11.3

Publisert 1. Mars 2023

Laget av Gerard Beekmans
Administrerende redaktør: Bruce Dubbs

Linux From Scratch: Versjon 11.3: Publisert 1. Mars 2023

by Laget av Gerard Beekmans and Administrerende redaktør: Bruce Dubbs

Table of Contents

Forord

Forord

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

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

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

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

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

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

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

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

--
Gerard Beekmans
gerard@linuxfromscratch.org

Publikum

Det er mange grunner til at du ønsker å lese denne boken. Et av spørsmålene mange 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 størrelseshensyn om ikke annet. Tenk på oppstartbar CDer, USB-pinner og innebygde systemer. Det er områder hvor LFS kan være gunstig.

En annen fordel med et spesialbygd Linux system er sikkerhet. Ved å kompilere hele systemet fra kildekoden, har du fullmakt til å revidere alt og bruk alle sikkerhetsoppdateringene du ønsker. Det er ikke lenger nødvendig å vente på at noen andre skal kompilere binære pakker som fikser et sikkerhetshull. Med mindre du undersøker oppdateringen og implementerer den selv, har du ingen garantier på at den nye binære pakken ble bygget riktig og løser 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 målarkitekturene til LFS er AMD/Intel x86 (32-bit) og x86_64 (64-bit) CPUer. På den annen side er instruksjonene i denne boken også kjent for å fungere, med noen modifikasjoner, med Power PC og ARM CPUer. Hovedforutsetningen for å bygge et system som bruker en av disse CPUene, i i tillegg til de på neste side, er et eksisterende Linux-system som f.eks tidligere LFS installasjon, Ubuntu, Red Hat/Fedora, SuSE eller annen distribusjon som retter seg mot arkitekturen du har. Vær også oppmerksom på at en 32-bit distribusjon kan installeres og brukes som et vertssystem på en 64-bit AMD/Intel datamaskin.

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, én gang for et 32-bitssystem og én gang for et 64-bitssystem. Dette støttes ikke direkte i LFS fordi det ville forstyrre pedagogisk mål om å gi instruksjonene som trengs for et enkelt grunnleggende Linux-system. Noen LFS/BLFS-redaktører opprettholder en forgrening av LFS for flerarkitektur, som er tilgjengelig på *https://www.linuxfromscratch.org/~thomas/multilib/index.html*. Men det er et avansert tema.

Forutsetninger

Å bygge et LFS system er ikke en enkel oppgave. Det krever en viss nivå av eksisterende kunnskap om Unix systemadministrasjon for å løse problemer og 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, Kjøretidsspråk og bildebehandling. I tillegg til generiske krav er det også arkitekturspesifikke krav. Det er også to områder for prøvebruk: Gtk3 og grafikk. LFS forsøker å samsvare med LSB spesifikasjoner for IA32 (32-bit x86) eller AMD64 (x86_64) arkitekturer omtalt i forrige avsnitt.

Note

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

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

Pakker levert av LFS som trengs for å tilfredsstille LSB kravene

LSB Kjerne:	Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux, Zlib

LSB Skrivebord:	Ingen
LSB Kjøretidsspråk:	Perl, Python
LSB Bildebehandling:	Ingen
LSB Gtk3 og LSB Grafikk (Prøvebruk):	Ingen

Pakker levert av BLFS som trengs for å tilfredsstille LSB kravene

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

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

LSB Kjerne:	Ingen
LSB Skrivebord:	Qt4 (men Qt5 er levert)
LSB Kjøretidsspråk:	Ingen
LSB Bildebehandling:	Ingen
LSB Gtk3 and LSB Grafikk (Prøvebruk):	Ingen

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 tilgangskontrolllister, som brukes til å definere mer finkornet skjønnsmessige tilgangsrettigheter for filer og kataloger.

- Attr

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

- Autoconf

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

- Automake

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

- Bash

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

- Bc

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

- Binutils

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

- 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.

- Eudev

  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.

- 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, Man-DB.

- Gettext

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

- Glibc

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

- GMP

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

- Gperf

  Denne pakken inneholder et program som genererer en perfekt hashfunksjon fra et nøkkelsett. Den kreves av Eudev .

- Grep

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

- Groff

  Denne pakken inneholder programmer for behandling og formatering av tekst. En viktig funksjon av disse programmene er å formatere 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 tastaturtabellfiler, 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 overførbar programmering på høyt nivå til ulike kallkonvensjoner. Noen programmer vet kanskje ikke på sammenstillingstidspunktet hvilke argumenter som skal overføres til en funksjon. For eksempel kan en tolk bli fortalt under kjøringen om antallet og typene argumenter som brukes til å kalle en gitt funksjon. Libffi kan brukes i slike programmer for å gi en bro fra tolkeprogrammet til kompilert kode.

- Libpipeline

  Libpipeline pakken inneholder et bibliotek for å manipulere kommandokøer av delprosesser på en fleksibel og praktisk måte. 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.

- Linux Kernel

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

- M4

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

- Make

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

- 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.

- Pkg-config

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

- Procps-NG

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

- Psmisc

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

- 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.

- Tar

  Denne pakken gir arkiverings- og utpakkingssmuligheter 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 tabulatortegn vil lage feil resultater.

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

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

Note

Vennligst konfigurer nettleseren din til å vise tekst med fast bredde og en god monospace" font-size="9pt font, så du kan skille tegnvariantene `Il1` eller `O0` 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 manualside (manside). Tallet innenfor parentes indikerer en bestemt del i manualene. For eksempel, **passwd** har to mansider. I henhold til LFS installasjonsinstruksjoner, vil disse to mansidene 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 mansiden den finner som stemmer med “passwd”, som vil bli `/usr/share/man/man1/passwd.1`. For dette eksemplet må du kjøre **man 5 passwd** for å lese siden som blir spesifisert. Merk at de fleste mansider ikke har duplikate sidenavn i forskjellige seksjoner. Derfor er, **man *<programnavn>*** generelt tilstrekkelig.

Struktur

Denne boken er delt inn i følgende deler.

Del I - Introduksjon

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

Del II - Forberedelse til bygging

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

Del III - Bygging av LFS kryssverktøykjede og midlertidige verktøy

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

Del IV - Bygge LFS systemet

Del IV guider leseren gjennom byggingen av LFS systemet—kompilere og installere alle pakkene én etter én, sette opp oppstartsskriptene og installere kjernen. Det resulterende Linux-systemet er grunnlaget som annen programvare kan bygges på, utvide systemet etter ønske. På slutten av denne boken er det en enkel 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/11.3/* 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 systemetet.

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.

Part I. Introduksjon

Chapter 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). Dette 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.

Som et alternativ til å installere en separat distribusjon på din maskin, ønsker du kanskje 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 side forrige utgivelse

I 11.3 utgivelsen, `--enable-default-pie` og `--enable-default-ssp` er aktivert for GCC. Disse teknikkene kan dempe noen ondsinnede angrep, men de gir ikke perfekt sikkerhet. Merk at noen lærebøker antar at disse alternativene er deaktivert, slik at hvis du kjører eksempler fra en slik lærebok på et LFS system, Må du kanskje deaktivere PIE og SSP med GCC alternativene `-fno-pie -no-pie -fno-stack-protection`.

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

Oppgradert til:

•

- Bash 5.2.15
- Bc 6.2.4
- Binutils-2.40
- Diffutils-3.9
- E2fsprogs-1.47.0
- Expat-2.5.0
- File-5.44
- Gawk-5.2.1
- Gettext-0.21.1
- Glibc-2.37
- Grep-3.8
- IANA-Etc-20230202
- Inetutils-2.4
- IPRoute2-6.1.0
- Less-608
- Libcap-2.67
- Libelf-0.188 (from elfutils)
- Libffi-3.4.4
- Linux-6.1.11
- Make-4.4
- Man-DB-2.11.2
- Man-pages-6.03
- Meson-1.0.0
- MPC-1.3.1
- MPFR-4.2.0
- Ncurses-6.4
- Ninja-1.11.1
- Openssl-3.0.8
- Procps-ng-4.0.2
- Psmisc-23.6
- Python-3.11.2
- Readline-8.2
- Sed-4.9
- Shadow-4.13
- SysVinit-3.06
- Tcl-8.6.13
- Texinfo-7.0.2

- Tzdata-2022g
- Vim-9.0.1273
- wheel-0.38.4
- XZ-Utils-5.4.1
- Zlib-1.2.13
- Zstd-1.5.4

Lagt til:

-
- grub-2.06-upstream_fixes-1.patch
- readline-8.2-upstream_fix-1.patch

Fjernet:

-
- zstd-1.5.2-upstream_fixes-1.patch

1.3. Endringslogg

Dette er versjon 11.3 av Linux From Scratch-boken, datert 1. Mars 2023. 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:

- 01.03.2023
  - [bdubbs] - LFS-11.3 utgitt.
- 19.02.2023
  - [xry111] - Bruk en oppdatering for GRUB for et problem utløst av e2fsprogs-1.47.0. Fikser *#5219*.
- 2023-02-13
  - [bdubbs] - Oppdatert til man-pages-6.03. Fikser *#5216*.
- 2023-02-11
  - [bdubbs] - Oppdatert til iana-etc-20230202. Adresserer *#5006*.
  - [bdubbs] - Oppdatert til zstd-1.5.4. Fikser *#5215*.
  - [bdubbs] - Oppdatert til Python3-3.11.2. Fikser *#5214*.
  - [bdubbs] - Oppdatert til e2fsprogs-1.47.0. Fikser *#5213*.
  - [bdubbs] - Oppdatert til linux-6.1.11. Fikser *#5210*.
  - [bdubbs] - Oppdatert til libcap-2.67. Fikser *#5209*.
  - [bdubbs] - Oppdatert til bc-6.2.4. Fikser *#5207*.
- 2023-02-07
  - [renodr] - Oppdatert til OpenSSL-3.0.8 (Sikkerhetsoppdatering). Fikser *#5211*.
  - [renodr] - Oppdatert til e2fsprogs-1.46.6 (Sikkerhetsoppdatering). Fikser *#5208*.

- 2023-02-02
  - [xry111] - Oppdatert til glibc-2.37. Fikser *#5203*.
  - [xry111] - Oppdatert til bc-6.2.3. Fikser *#5204*.
  - [xry111] - Oppdatert til linux-6.1.9. Fikser *#5205*.
  - [xry111] - Oppdatert til vim-9.0.1273. Adresserer *#4500*.
  - [xry111] - Fjerner `--disable-exec-static-tramp` for libffi.
- 2023-02-01
  - [bdubbs] - Oppdatert til texinfo-7.0.2. Fikser *#5202*.
  - [bdubbs] - Oppdatert til linux-6.1.8. Fikser *#5201*.
  - [bdubbs] - Oppdatert til diffutils-3.9. Fikser *#5199*.
- 2023-01-15
  - [thomas] - Legg til libsframe til online_usrlib i stripping. libsframe.so.0.0.0 er i bruk av strip.
  - [bdubbs] - Oppdatert til iana-etc-20230109. Adresserer *#5006*.
  - [bdubbs] - Oppdatert til binutils-2.40. Fikser *#5198*.
  - [bdubbs] - Oppdatert til bc-6.2.2. Fikser *#5192*.
  - [bdubbs] - Oppdatert til linux-6.1.6. Fikser *#5193*.
  - [bdubbs] - Oppdatert til man-db-2.11.2. Fikser *#5196*.
  - [bdubbs] - Oppdatert til mpfr-4.2.0. Fikser *#5195*.
  - [bdubbs] - Oppdatert til ncurses-6.4. Fikser *#5194*.
  - [bdubbs] - Oppdatert til xz-5.4.1. Fikser *#5197*.
- 2023-01-01
  - [thomas] - Fjerner en foreldet sed fra mpc.
- 2022-12-31
  - [bdubbs] - Oppdatert til iana-etc-20221220. Adresserer *#5006*.
  - [bdubbs] - Oppdatert til sysvinit-3.06. Fikser *#5186*.
  - [bdubbs] - Oppdatert til mpc-1.3.1. Fikser *#5185*.
  - [bdubbs] - Oppdatert til meson-1.0.0. Fikser *#5190*.
  - [bdubbs] - Oppdatert til man-pages-6.02. Fikser *#5188*.
  - [bdubbs] - Oppdatert til linux-6.1.1. Fikser *#5179*.
  - [bdubbs] - Oppdatert til file-5.44. Fikser *#5191*.
  - [bdubbs] - Oppdatert til bc-6.2.1. Fikser *#5189*.
- 2022-12-15
  - [bdubbs] - Sørg for at en gawk hardlenke er oppdatert i Kapittel 8. Fikser *#5180*.
  - [bdubbs] - Oppdatert til iana-etc-20221209. Adresserer *#5006*.
  - [bdubbs] - Oppdatert til vim-9.0.1060. Adresserer *#4500*.
  - [bdubbs] - Oppdatert til iproute2-6.1.0. Fikser *#5184*.

- [bdubbs] - Oppdatert til xz-5.4.0. Fikser *#5183*.
- [bdubbs] - Oppdatert til bash-5.2.15. Fikser *#5182*.
- [bdubbs] - Oppdatert til psmisc-23.6. Fikser *#5181*.
- [bdubbs] - Oppdatert til mpc-1.3.0. Fikser *#5178*.
- [bdubbs] - Oppdatert til python3-3.11.1. Fikser *#5177*.
- [bdubbs] - Oppdatert til procps-ng-4.0.2. Fikser *#5176*.

- 2022-12-01
  - [bdubbs] - Oppdatert til linux-6.0.11 (Sikkerhetsoppdatering). Fikser *#5175*.
- 2022-12-01
  - [bdubbs] - Oppdatert til iana-etc-20221122. Adresserer *#5006*.
  - [bdubbs] - Oppdatert til xz-5.2.9. Fikser *#5174*.
  - [bdubbs] - Oppdatert til tzdata-2022g. Fikser *#5172*.
  - [bdubbs] - Oppdatert til texinfo-7.0.1. Fikser *#5173*.
  - [bdubbs] - Oppdatert til tcl-8.6.13. Fikser *#5170*.
  - [bdubbs] - Oppdatert til meson-0.64.1. Fikser *#5169*.
  - [bdubbs] - Oppdatert til linux-6.0.10. Fikser *#5171*.
  - [bdubbs] - Oppdatert til gawk-5.2.1. Fikser *#5168*.
- 2022-11-22
  - [xry111] - Oppdatert til linux-6.0.9. Fikser *#5162*.
  - [xry111] - Oppdatert til libpipeline-1.5.7. Fikser *#5163*.
  - [xry111] - Oppdatert til xz-5.2.8. Fikser *#5164*.
  - [xry111] - Oppdatert til man-db-2.11.1. Fikser *#5166*.
  - [xry111] - Oppdatert til mpfr-4.1.1. Fikser *#5167*.
  - [xry111] - Slutt å deaktivere desimalfly for midlertidig GCC, så mpfr vil bli bygget med støtte for desimalflyt.
  - [xry111] - Oppdater instruksjonen for wheel for å unngå å stole på utdaterte Python funksjoner.
- 2022-11-10
  - [bdubbs] - Fikser make-4.4 bug. Fikser *#5160*.
  - [bdubbs] - Oppdatert til wheel-0.38.4 (Python Modul). Fikser *#5155*.
  - [bdubbs] - Oppdatert til texinfo-7.0. Fikser *#5159*.
  - [bdubbs] - Oppdatert til sysvinit-3.05. Fikser *#5153*.
  - [bdubbs] - Oppdatert til shadow-4.13. Fikser *#5161*.
  - [bdubbs] - Oppdatert til sed-4.9. Fikser *#5157*.
  - [bdubbs] - Oppdatert til meson-0.64.0. Fikser *#5156*.
  - [bdubbs] - Oppdatert til linux-6.0.7. Fikser *#5154*.
  - [bdubbs] - Oppdatert til elfutils-0.188. Fikser *#5152*.
  - [bdubbs] - Oppdatert til bc-6.1.1. Fikser *#5151*.

- [bdubbs] - Oppdatert til bash-5.2.9. Fikser *#5158*.

- 2022-11-01
  - [bdubbs] - Oppdatert til openssl-3.0.7 (Sikkerhetsoppdatering). Fikser *#5132*.
  - [bdubbs] - Oppdatert til iana-etc-20221025. Adresserer *#5006*.
  - [bdubbs] - Oppdatert til tzdata-2022f. Fikser *#5148*.
  - [bdubbs] - Oppdatert til Python3-3.11.0. Fikser *#5145*.
  - [bdubbs] - Oppdatert til procps-ng-4.0.1. Fikser *#5141*.
  - [bdubbs] - Oppdatert til man-pages-6.01. Fikser *#5140*.
  - [bdubbs] - Oppdatert til man-db-2.11.0. Fikser *#5139*.
  - [bdubbs] - Oppdatert til make-4.4. Fikser *#5149*.
  - [bdubbs] - Oppdatert til linux-6.0.6. Fikser *#5142*.
  - [bdubbs] - Oppdatert til libffi-3.4.4. Fikser *#5144*.
  - [bdubbs] - Oppdatert til inetutils-2.4. Fikser *#5147*.
  - [bdubbs] - Oppdatert til expat-2.5.0. Fikser *#5132*.
- 2022-10-17
  - [bdubbs] - Oppdatert til linux-6.0.2 (Sikkerhetsoppdatering). Fikser *#5138*.
- 2022-10-15
  - [bdubbs] - Oppdatert til iana-etc-20221007. Adresserer *#5006*.
  - [bdubbs] - Oppdatert til vim-9.0.0739. Adresserer *#5006*.
  - [bdubbs] - Lagt til oppstrøms oppdateringer til readline og bash. Fikser *#5131*.
  - [bdubbs] - Oppdatert til zlib-1.2.13. Fikser *#5137*.
  - [bdubbs] - Oppdatert til man-pages-6.00. Fikser *#5136*.
  - [bdubbs] - Oppdatert til gettext-0.21.1. Fikser *#5130*.
  - [bdubbs] - Oppdatert til iproute2-6.0.0. Fikser *#5127*.
  - [bdubbs] - Oppdatert til meson-0.63.3. Fikser *#5129*.
  - [bdubbs] - Oppdatert til Python-3.10.8. Fikser *#5133*.
  - [bdubbs] - Oppdatert til xz-5.2.7. Fikser *#5133*.
  - [bdubbs] - Oppdatert til tzdata-2022e. Fikser *#5134*.
  - [bdubbs] - Oppdatert til linux-6.0.1. Fikser *#5135*.
- 2022-10-04
  - [renodr] - Oppdatert til Linux-5.19.13. Hvis du bruker en Intel GPU på en bærbar PC, vennligst oppdater fra Linux-5.19.12 umiddelbart for å forhindre skade på skjermen. Fikser *#5125*.
- 2022-10-01
  - [bdubbs] - Oppdatert til iana-etc-20220922. Adresserer *#5006*.
  - [bdubbs] - Oppdatert til tzdata-2022d. Fikser *#5119*.
  - [bdubbs] - Oppdatert til readline-8.2. Fikser *#5121*.

- [bdubbs] - Oppdatert til linux-5.19.12. Fikser *#5115*.
- [bdubbs] - Oppdatert til libffi-3.4.3. Fikser *#5116*.
- [bdubbs] - Oppdatert til libcap-2.66. Fikser *#5120*.
- [bdubbs] - Oppdatert til bc-6.0.4. Fikser *#5114*.
- [bdubbs] - Oppdatert til bash-5.2. Fikser *#5122*.

- 2022-09-22
  - [bdubbs] - Oppdatert til expat-2.4.9 (Sikkerhetsoppdatering). Fikser *#5117*.
- 2022-09-20
  - [bdubbs] - Tilpass instruksjoner avhengig av vertsoppsett av /dev/shm når du oppretter virtuelle filsystemer for chroot.
- 2022-09-15
  - [bdubbs] - Oppdatert til file-5.43. Fikser *#5113*.
  - [bdubbs] - Oppdatert til linux-5.19.8. Fikser *#5111*.
  - [bdubbs] - Oppdatert til gawk-5.2.0. Fikser *#5108*.
  - [bdubbs] - Oppdatert til meson-0.63.2. Fikser *#5106*.
  - [bdubbs] - Oppdatert til ninja-1.11.1. Fikser *#5103*.
  - [bdubbs] - Oppdatert til bc-6.0.2. Fikser *#5102*.
  - [bdubbs] - Fiks plasseringen av udev regler i eudev. Fikser *#5112*.
  - [bdubbs] - Fjerner en advarsel for egrep og fgrep som gjør at tester for noen pakker mislykkes.
  - [bdubbs] - Slett en tom binutils manside. Fikser *#5100*.
- 2022-09-10
  - [pierre] - Lagt til `--enable-default-pie` og `--enable-default-ssp` til byggingen av GCC. Begrunnelse og noen rapporter på *#5107*.
- 2022-09-07
  - [bdubbs] - Oppdatert til shadow-4.12.3. Fikser *#5101*.
  - [bdubbs] - Oppdatert til Python3-3.10.7. Fikser *#5109*.
  - [bdubbs] - Oppdatert til linux-5.19.7. Fikser *#5099*.
  - [bdubbs] - Oppdatert til less-608. Fikser *#5104*.
  - [bdubbs] - Oppdatert til grep-3.8. Fikser *#5105*.
- 2022-09-01
  - [bdubbs] - LFS-11.2 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

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

1.5.1. Ting å nevne

Bortsett fra en kort forklaring av problemet som oppleves, enhver forespørsel om hjelp bør inkludere disse viktige tingene:

- Versjonen av boken som brukes (i dette tilfellet 11.3)
- Vertsdistribusjonen og versjonen som brukes til å lage LFS
- Utdata fra Host System Requirements 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

Note

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

1.5.2. Konfigurasjonsskript problemer

Hvis noe går galt mens du kjører **configure** 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 -DALIASPATH=\"/mnt/lfs/usr/share/locale:.\"
-DLOCALEDIR=\"/mnt/lfs/usr/share/locale\"
-DLIBDIR=\"/mnt/lfs/usr/lib\"
-DINCLUDEDIR=\"/mnt/lfs/usr/include\" -DHAVE_CONFIG_H -I. -I.
-g -O2 -c getopt1.c
gcc -g -O2 -static -o make ar.o arscan.o commands.o dir.o
expand.o file.o function.o getopt.o implicit.o job.o main.o
misc.o read.o remake.o rule.o signame.o variable.o vpath.o
default.o remote-stub.o version.o opt1.o
-lutil job.o: In function `load_too_high':
/lfs/tmp/make-3.79.1/job.c:1565: undefined reference
to `getloadavg'
collect2: ld returned 1 exit status
make[2]: *** [make] Error 1
make[2]: Leaving directory `/lfs/tmp/make-3.79.1'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/lfs/tmp/make-3.79.1'
make: *** [all-recursive-am] Error 2
```

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

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

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

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

Part II. Forbereder for byggingen

Chapter 2. Klargjøring av vertssystemet

2.1. Introduksjon

I dette kapittelet, vertsverktøyene som trengs for å bygge LFS kontrolleres og om nødvendig installeres. Deretter vil en partisjon klargjøres som vert for LFS systemet. Vi lager partisjonen, 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 "<pakke-navn>-devel" eller "<pakke-navn>-dev". Pass på å installere disse hvis distribusjonen din har 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 2.40 anbefales ikke ettersom de ikke har blitt testet)
- **Bison-2.7** (/usr/bin/yacc bør være en lenke til bison eller et lite skript som kjører bison)
- **Coreutils-6.9**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk bør være en link til gawk)
- **GCC-5.1** inkludert C++ kompilatoren, **g++** (Versjoner større enn 12.2.0 er ikke anbefalt da de ikke er testet). C og C++ standard biblioteker (med deklarasjoner) må også være tilstede slik at C++ kompilatoren kan bygge vertsbaserte programmer
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Linux Kernel-3.2**

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

  Hvis vertskjernen er tidligere enn 3.2 du må erstatte kjernen med en mer oppdatert versjon. Det er to måter du kan gjøre dette på. Først, se om din Linux leverandør tilbyr en 3.2 eller senere kjernepakke. I så fall kan det være lurt å installere den. Hvis din leverandøren ikke tilbyr en akseptabel kjernepakke, eller du foretrekker å la være å installer den, kan du kompilere en kjerne selv. Instruksjoner for å kompilere kjernen og konfigurere oppstartslasteren (forutsatt at verten bruker GRUB) er lokalisert i Kapittel 10.
- **M4-1.4.10**

- **Make-4.0**
- **Patch-2.5.4**
- **Perl-5.8.8**
- **Python-3.4**
- **Sed-4.1.5**
- **Tar-1.22**
- **Texinfo-4.7**
- **Xz-5.0.0**

Important

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

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

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

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

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

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

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

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

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

bash version-check.sh
```

2.3. Bygge LFS i etapper

LFS er designet for å bygges i én økt. Det er det instruksjonene forutsetter, at systemet ikke vil bli slått av under prosessen. Det betyr ikke at 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, være sikker på én 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 kompileringsinstruksjoner 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 “Skifte eierskap” 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`, kjøre kommandoene i Section 7.3.1, “Montering og fylling av /dev” og Section 7.3.2, “Montering av det virtuelle kjernefilsystemer”.

2.4. Opprette en ny partisjon

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

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

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

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

Note

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

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

2.4.1. Andre partisjonsproblemer

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

2.4.1.1. Rotpartisjonen

En 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 før den slår av maskinen. I dette tilfellet bør størrelsen på vekselminnepartisjonen være minst like stor som systemets installerte RAM.

Bruk av vekselminne er aldri bra. For mekaniske harddisker kan du generelt fortelle om et system veksler ved å bare lytte til diskaktivitet og observere hvordan systemet reagerer på kommandoer. 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.

Note

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 oppstartsdisken.

2.4.1.4. Bekvemmelig partisjoner

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

- /boot – Sterkt anbefalt. Bruk denne partisjonen til å lagre kjerner og annen oppstartsinformasjon. For å minimere potensielle oppstartsproblemer med større disker, 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 én 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 mappe 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 Section 10.2, “Opprette /etc/fstab filen”.

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, ReiserFS, 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

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 velg 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.

Caution

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

```
echo $LFS
```

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

Note

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

En annen vurdering er metoden som brukes for å logge på vertssystemet. Hvis du logger på via en grafisk skjermbehandler, brukerens `.bash_profile` brukes vanligvis ikke når en virtuell terminal startes. I dette tilfellet legger du til export kommandoen til filen `.bashrc` for brukeren og `root`. I tillegg, noen distribusjoner bruk en "if" test, og kjører ikke de resterende `.bashrc` instruksjoner for en ikke-interaktiv bash påkallelse. Pass på å plassere export 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.

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.

Warning

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.

Chapter 3. Pakker og oppdateringer

3.1. Introduksjon

Dette kapittelet inneholder en liste over pakker som må lastes ned for å bygge et grunnleggende Linux system. De oppførte versjonsnumrene tilsvarer versjoner av programvaren som er kjent for å fungere, og denne boken er basert på deres bruk. Vi anbefaler på det sterkeste å ikke bruke forskjellige versjoner fordi konstruksjonens kommandoer for 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 løsninger. Disse løsningene vil bli utviklet og stabilisert i utviklingsversjon av boken.

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

Nedlastingsplasseringer er kanskje ikke alltid tilgjengelige. Hvis en 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 skrivetillatelse på en mappe, er det bare eieren av en fil som kan slette filen i en låst mappe. Følgende kommando vil aktivere skrive og låste moduser:

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

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

Note

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 én 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 idé 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 (2.3.1) - 348 KB:
Hjemmeside: *https://savannah.nongnu.org/projects/acl*
Nedlasting: *https://download.savannah.gnu.org/releases/acl/acl-2.3.1.tar.xz*
MD5 sum: `95ce715fe09acca7c12d3306d0f076b2`

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

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

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

• Bash (5.2.15) - 10,695 KB:
Hjemmeside: *https://www.gnu.org/software/bash/*
Nedlasting: *https://ftp.gnu.org/gnu/bash/bash-5.2.15.tar.gz*
MD5 sum: `4281bb43497f3905a308430a8d6a30a5`

• Bc (6.2.4) - 447 KB:
Hjemmeside: *https://git.gavinhoward.com/gavin/bc*
Nedlasting: *https://github.com/gavinhoward/bc/releases/download/6.2.4/bc-6.2.4.tar.xz*
MD5 sum: `5245ff400df17b66be7621c7a6498953`

• Binutils (2.40) - 24,650 KB:
Hjemmeside: *https://www.gnu.org/software/binutils/*
Nedlasting: *https://sourceware.org/pub/binutils/releases/binutils-2.40.tar.xz*
MD5 sum: `007b59bd908a737c06e5a8d3d2c737eb`

• 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: `c28f119f405a2304ff0a7ccdcc629713`

• 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: `50fcafcecde5a380415b12e9c574e0b2`

• Coreutils (9.1) - 5,570 KB:
Hjemmeside: *https://www.gnu.org/software/coreutils/*
Nedlasting: *https://ftp.gnu.org/gnu/coreutils/coreutils-9.1.tar.xz*
MD5 sum: `8b1ca4e018a7dce9bb937faec6618671`

• 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 (3.9) - 1,551 KB:
Hjemmeside: *https://www.gnu.org/software/diffutils/*
Nedlasting: *https://ftp.gnu.org/gnu/diffutils/diffutils-3.9.tar.xz*
MD5 sum: `cf0a65266058bf22fe3feb69e57ffc5b`

• E2fsprogs (1.47.0) - 9,412 KB:
Hjemmeside: *http://e2fsprogs.sourceforge.net/*
Nedlasting: *https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.47.0/e2fsprogs-1.47.0.tar.gz*
MD5 sum: `6b4f18a33873623041857b4963641ee9`

• Elfutils (0.188) - 8,900 KB:
Hjemmeside: *https://sourceware.org/elfutils/*
Nedlasting: *https://sourceware.org/ftp/elfutils/0.188/elfutils-0.188.tar.bz2*
MD5 sum: `efb25a91873b2eec4df9f31e6a4f4e5c`

• Eudev (3.2.11) - 2,075 KB:
Nedlasting: *https://github.com/eudev-project/eudev/releases/download/v3.2.11/eudev-3.2.11.tar.gz*
MD5 sum: `417ba948335736d4d81874fba47a30f7`

• Expat (2.5.0) - 450 KB:
Hjemmeside: *https://libexpat.github.io/*
Nedlasting: *https://prdownloads.sourceforge.net/expat/expat-2.5.0.tar.xz*
MD5 sum: `ac6677b6d1b95d209ab697ce8b688704`

• 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 (5.44) - 1,159 KB:
Hjemmeside: *https://www.darwinsys.com/file/*
Nedlasting: *https://astron.com/pub/file/file-5.44.tar.gz*
MD5 sum: `a60d586d49d015d842b9294864a89c7a`

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

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

• Gawk (5.2.1) - 3,332 KB:
Hjemmeside: *https://www.gnu.org/software/gawk/*
Nedlasting: *https://ftp.gnu.org/gnu/gawk/gawk-5.2.1.tar.xz*
MD5 sum: `02956bc5d117a7437bb4f7039f23b964`

• GCC (12.2.0) - 82,662 KB:
Hjemmeside: *https://gcc.gnu.org/*
Nedlasting: *https://ftp.gnu.org/gnu/gcc/gcc-12.2.0/gcc-12.2.0.tar.xz*
MD5 sum: `73bafd0af874439dcdb9fc063b6fb069`
SHA256 sum:

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

• Gettext (0.21.1) - 9,819 KB:
Hjemmeside: *https://www.gnu.org/software/gettext/*
Nedlasting: *https://ftp.gnu.org/gnu/gettext/gettext-0.21.1.tar.xz*
MD5 sum: `27fcc8a42dbc8f334f23a08f1f2fe00a`

• Glibc (2.37) - 18,244 KB:
Hjemmeside: *https://www.gnu.org/software/libc/*
Nedlasting: *https://ftp.gnu.org/gnu/glibc/glibc-2.37.tar.xz*
MD5 sum: `e89cf3dcb64939d29f04b4ceead5cc4e`

Note

Glibc utviklerne opprettholder en *Git branch* som inneholder oppdateringer som anses verdig Glibc-2.37 men utviklet seg dessverre etter Glibc-2.37 utgivelsen. LFS redaksjonen vil utstede en sikkerhetsrådgivning hvis noen sikkerhetsfiks 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 (6.2.1) - 1,980 KB:
Hjemmeside: *https://www.gnu.org/software/gmp/*
Nedlasting: *https://ftp.gnu.org/gnu/gmp/gmp-6.2.1.tar.xz*
MD5 sum: `0b82665c4a92fd2ade7440c13fcaa42b`

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

• Grep (3.8) - 1,670 KB:
Hjemmeside: *https://www.gnu.org/software/grep/*
Nedlasting: *https://ftp.gnu.org/gnu/grep/grep-3.8.tar.xz*
MD5 sum: `dc6e4d18d4659e6e7552fc4a183c8ac9`

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

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

• Gzip (1.12) - 807 KB:
Hjemmeside: *https://www.gnu.org/software/gzip/*
Nedlasting: *https://ftp.gnu.org/gnu/gzip/gzip-1.12.tar.xz*
MD5 sum: `9608e4ac5f061b2a6479dc44e917a5db`

• Iana-Etc (20230202) - 586 KB:
Hjemmeside: *https://www.iana.org/protocols*
Nedlasting: *https://github.com/Mic92/iana-etc/releases/download/20230202/iana-etc-20230202.tar.gz*
MD5 sum: `e64685d046cd0dfe94b5c66e294cf9ef`

• Inetutils (2.4) - 1,522 KB:
Hjemmeside: *https://www.gnu.org/software/inetutils/*
Nedlasting: *https://ftp.gnu.org/gnu/inetutils/inetutils-2.4.tar.xz*
MD5 sum: `319d65bb5a6f1847c4810651f3b4ba74`
SHA256 sum:

• 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 (6.1.0) - 885 KB:
Hjemmeside: *https://www.kernel.org/pub/linux/utils/net/iproute2/*
Nedlasting: *https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-6.1.0.tar.xz*
MD5 sum: `f3ff4461e25dbc5ef1fb7a9167a9523d`

• Kbd (2.5.1) - 1,457 KB:
Hjemmeside: *https://kbd-project.org/*
Nedlasting: *https://www.kernel.org/pub/linux/utils/kbd/kbd-2.5.1.tar.xz*
MD5 sum: `10f10c0a9d897807733f2e2419814abb`

• Kmod (30) - 555 KB:
Nedlasting: *https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-30.tar.xz*
MD5 sum: `85202f0740a75eb52f2163c776f9b564`

• Less (608) - 354 KB:
Hjemmeside: *https://www.greenwoodsoftware.com/less/*
Nedlasting: *https://www.greenwoodsoftware.com/less/less-608.tar.gz*
MD5 sum: `1cdec714569d830a68f4cff11203cdba`

• LFS-Bootscripts (20230101) - 33 KB:
Nedlasting: *https://www.linuxfromscratch.org/lfs/downloads/11.3/lfs-bootscripts-20230101.tar.xz*
MD5 sum: `569217b0b56f98fd267d38d72ee20132`

• Libcap (2.67) - 183 KB:
Hjemmeside: *https://sites.google.com/site/fullycapable/*
Nedlasting: *https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.67.tar.xz*
MD5 sum: `06333f4301657298890fd8d6f1fb4793`

• Libffi (3.4.4) - 1,331 KB:
Hjemmeside: *https://sourceware.org/libffi/*
Nedlasting: *https://github.com/libffi/libffi/releases/download/v3.4.4/libffi-3.4.4.tar.gz*
MD5 sum: `0da1a5ed7786ac12dcbaf0d499d8a049`

• Libpipeline (1.5.7) - 956 KB:
Hjemmeside: *https://libpipeline.nongnu.org/*
Nedlasting: *https://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.7.tar.gz*
MD5 sum: `1a48b5771b9f6c790fb4efdb1ac71342`

• Libtool (2.4.7) - 996 KB:
Hjemmeside: *https://www.gnu.org/software/libtool/*
Nedlasting: *https://ftp.gnu.org/gnu/libtool/libtool-2.4.7.tar.xz*
MD5 sum: `2fc0b6ddcd66a89ed6e45db28fa44232`

• Linux (6.1.11) - 131,653 KB:
Hjemmeside: *https://www.kernel.org/*
Nedlasting: *https://www.kernel.org/pub/linux/kernel/v6.x/linux-6.1.11.tar.xz*
MD5 sum: `f91621912cd58ac6d4128d4057980e7d`

Note

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.

• 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) - 2,254 KB:
Hjemmeside: *https://www.gnu.org/software/make/*
Nedlasting: *https://ftp.gnu.org/gnu/make/make-4.4.tar.gz*
MD5 sum: `d7575a26a94ee8427130e9db23cdaa78`

• Man-DB (2.11.2) - 1,908 KB:
Hjemmeside: *https://www.nongnu.org/man-db/*
Nedlasting: *https://download.savannah.gnu.org/releases/man-db/man-db-2.11.2.tar.xz*
MD5 sum: `a7d59fb2df6158c44f8f7009dcc6d875`

• Man-pages (6.03) - 2,134 KB:
Hjemmeside: *https://www.kernel.org/doc/man-pages/*
Nedlasting: *https://www.kernel.org/pub/linux/docs/man-pages/man-pages-6.03.tar.xz*
MD5 sum: `c62b7c944bb0887a35edab7cab301357`

• Meson (1.0.0) - 2,051 KB:
Hjemmeside: *https://mesonbuild.com*
Nedlasting: *https://github.com/mesonbuild/meson/releases/download/1.0.0/meson-1.0.0.tar.gz*
MD5 sum: `009b78125467cd9ee4d467175a5c12e1`

• 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 (4.2.0) - 1,443 KB:
Hjemmeside: *https://www.mpfr.org/*
Nedlasting: *https://ftp.gnu.org/gnu/mpfr/mpfr-4.2.0.tar.xz*
MD5 sum: `a25091f337f25830c16d2054d74b5af7`

• Ncurses (6.4) - 3,528 KB:
Hjemmeside: *https://www.gnu.org/software/ncurses/*
Nedlasting: *https://invisible-mirror.net/archives/ncurses/ncurses-6.4.tar.gz*
MD5 sum: `5a62487b5d4ac6b132fe2bf9f8fad29b`

• Ninja (1.11.1) - 225 KB:
Hjemmeside: *https://ninja-build.org/*
Nedlasting: *https://github.com/ninja-build/ninja/archive/v1.11.1/ninja-1.11.1.tar.gz*
MD5 sum: `32151c08211d7ca3c1d832064f6939b0`

• OpenSSL (3.0.8) - 14,800 KB:
Hjemmeside: *https://www.openssl.org/*
Nedlasting: *https://www.openssl.org/source/openssl-3.0.8.tar.gz*
MD5 sum: `61e017cf4fea1b599048f621f1490fbd`

• 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 (5.36.0) - 12,746 KB:
Hjemmeside: *https://www.perl.org/*
Nedlasting: *https://www.cpan.org/src/5.0/perl-5.36.0.tar.xz*
MD5 sum: `826e42da130011699172fd655e49cfa2`

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

• Procps (4.0.2) - 1250 KB:
Hjemmeside: *https://sourceforge.net/projects/procps-ng*
Nedlasting: *https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-4.0.2.tar.xz*
MD5 sum: `691748c4767f19b9d94ed9d088e40c4d`

• Psmisc (23.6) - 415 KB:
Hjemmeside: *https://gitlab.com/psmisc/psmisc*
Nedlasting: *https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.6.tar.xz*
MD5 sum: `ed3206da1184ce9e82d607dc56c52633`

• Python (3.11.2) - 19,428 KB:
Hjemmeside: *https://www.python.org/*
Nedlasting: *https://www.python.org/ftp/python/3.11.2/Python-3.11.2.tar.xz*
MD5 sum: `a957cffb58a89303b62124896881950b`

• Python Documentation (3.11.2) - 7,598 KB:
Nedlasting: *https://www.python.org/ftp/python/doc/3.11.2/python-3.11.2-docs-html.tar.bz2*
MD5 sum: `eb4132c780b60b5782a4f66b29b08d5c`

• Readline (8.2) - 2,973 KB:
Hjemmeside: *https://tiswww.case.edu/php/chet/readline/rltop.html*
Nedlasting: *https://ftp.gnu.org/gnu/readline/readline-8.2.tar.gz*
MD5 sum: `4aa1b31be779e6b84f9a96cb66bc50f6`

• 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`

• Shadow (4.13) - 1,722 KB:
Hjemmeside: *https://shadow-maint.github.io/shadow/*
Nedlasting: *https://github.com/shadow-maint/shadow/releases/download/4.13/shadow-4.13.tar.xz*
MD5 sum: `b1ab01b5462ddcf43588374d57bec123`

• Sysklogd (1.5.1) - 88 KB:
Hjemmeside: *https://www.infodrom.org/projects/sysklogd/*
Nedlasting: *https://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.1.tar.gz*
MD5 sum: `c70599ab0d037fde724f7210c2c8d7f8`

• Sysvinit (3.06) - 247 KB:
Hjemmeside: *https://savannah.nongnu.org/projects/sysvinit*
Nedlasting: *https://github.com/slicer69/sysvinit/releases/download/3.06/sysvinit-3.06.tar.xz*
MD5 sum: `96771d0a88315c91199830ea49b859ca`

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

• Tcl (8.6.13) - 10,581 KB:
Hjemmeside: *http://tcl.sourceforge.net/*
Nedlasting: *https://downloads.sourceforge.net/tcl/tcl8.6.13-src.tar.gz*
MD5 sum: `0e4358aade2f5db8a8b6f2f6d9481ec2`

• Tcl Documentation (8.6.13) - 1,165 KB:
Nedlasting: *https://downloads.sourceforge.net/tcl/tcl8.6.13-html.tar.gz*
MD5 sum: `4452f2f6d557f5598cca17b786d6eb68`

• Texinfo (7.0.2) - 4,762 KB:
Hjemmeside: *https://www.gnu.org/software/texinfo/*
Nedlasting: *https://ftp.gnu.org/gnu/texinfo/texinfo-7.0.2.tar.xz*
MD5 sum: `be9500f3a361525622850ecb1b1fc024`

• Time Zone Data (2022g) - 430 KB:
Hjemmeside: *https://www.iana.org/time-zones*
Nedlasting: *https://www.iana.org/time-zones/repository/releases/tzdata2022g.tar.gz*
MD5 sum: `884250fd2a8a55f6322900ad4ab94d7b`

• Udev-lfs Tarball (udev-lfs-20171102) - 11 KB:
Nedlasting: *https://anduin.linuxfromscratch.org/LFS/udev-lfs-20171102.tar.xz*
MD5 sum: `27cd82f9a61422e186b9d6759ddf1634`

• Util-linux (2.38.1) - 7,321 KB:
Hjemmeside: *https://git.kernel.org/pub/scm/utils/util-linux/util-linux.git/*
Nedlasting: *https://www.kernel.org/pub/linux/utils/util-linux/v2.38/util-linux-2.38.1.tar.xz*
MD5 sum: `cd11456f4ddd31f7fbfdd9488c0c0d02`

• Vim (9.0.1273) - 10,892 KB:
Hjemmeside: *https://www.vim.org*
Nedlasting: *https://anduin.linuxfromscratch.org/LFS/vim-9.0.1273.tar.xz*
MD5 sum: `9c80755d2d95ec4ef713f66e57671797`

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

• Wheel (0.38.4) - 66 KB:
Hjemmeside: *https://pypi.org/project/wheel/*
Nedlasting: *https://pypi.org/packages/source/w/wheel/wheel-0.38.4.tar.gz*
MD5 sum: `83bb4e7bd4d687d398733f341a64ab91`

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

• Xz Utils (5.4.1) - 1,451 KB:
Hjemmeside: *https://tukaani.org/xz*
Nedlasting: *https://tukaani.org/xz/xz-5.4.1.tar.xz*
MD5 sum: `47d831c659e94071d5dd384d0d1ed4c6`

• Zlib (1.2.13) - 1267 KB:
Hjemmeside: *https://www.zlib.net/*
Nedlasting: *https://zlib.net/zlib-1.2.13.tar.xz*
MD5 sum: `7d9fc1d78ae2fa3e84fe98b77d006c63`

• Zstd (1.5.4) - 2,111 KB:
Hjemmeside: *https://facebook.github.io/zstd/*
Nedlasting: *https://github.com/facebook/zstd/releases/download/v1.5.4/zstd-1.5.4.tar.gz*
MD5 sum: `2352b1f9ccc7446641046bb3d440c3ed`

Total størrelse på disse pakkene: ca 462 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/11.3/bzip2-1.0.8-install_docs-1.patch*
MD5 sum: `6a5ac7e89b791aae556de0f745916f7f`

• Coreutils Internationalization Fixes Patch - 166 KB:
Nedlasting: *https://www.linuxfromscratch.org/patches/lfs/11.3/coreutils-9.1-i18n-1.patch*
MD5 sum: `c1ac7edf095027460716577633da9fc5`

• Glibc FHS Patch - 2.8 KB:
Nedlasting: *https://www.linuxfromscratch.org/patches/lfs/11.3/glibc-2.37-fhs-1.patch*
MD5 sum: `9a5997c3452909b1769918c759eff8a2`

• GRUB Upstream Fixes Patch - 8 KB:
Nedlasting: *https://www.linuxfromscratch.org/patches/lfs/11.3/grub-2.06-upstream_fixes-1.patch*
MD5 sum: `da388905710bb4cbfbc7bd7346ff9174`

• Kbd Backspace/Delete Fix Patch - 12 KB:
Nedlasting: *https://www.linuxfromscratch.org/patches/lfs/11.3/kbd-2.5.1-backspace-1.patch*
MD5 sum: `f75cca16a38da6caa7d52151f7136895`

• Readline Upstream Fix Patch - 1.3 KB:
Nedlasting: *https://www.linuxfromscratch.org/patches/lfs/11.3/readline-8.2-upstream_fix-1.patch*
MD5 sum: `dd1764b84cfca6b677f44978218a75da`

• Sysvinit Consolidated Patch - 2.5 KB:
Nedlasting: *https://www.linuxfromscratch.org/patches/lfs/11.3/sysvinit-3.06-consolidated-1.patch*
MD5 sum: `17ffccbb8e18c39e8cedc32046f3a475`

Total størrelse på disse oppdateringene: ca 194.2 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.

Chapter 4. Siste forberedelser

4.1. Introduksjon

I dette kapittelet vil vi utføre noen tilleggsoppgaver for å forberede byggingen av det midlertidige systemet. Vi vil lage et sett med mapper i `$LFS` (hvor vi vil installere midlertidige verktøy, legg til en uprivilegert bruker, og skape et passende byggemiljø for denne brukeren. Det vil også 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 kompileres 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
```

Note

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

Note

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`

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

`LFS=/mnt/lfs`

`LFS` variabelen skal settes til det valgte monteringenspunktet.

`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.

Important

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, kjør:

```
[ ! -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 Section 8.34, “Bash-5.2.15” ikke er konfigurert til å laste eller kjøre `/etc/bash.bashrc`, så denne filen er ubrukelig på et fullført LFS system.

Til slutt, å ha miljøet fullt forberedt for å bygge midlertidige verktøy, 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 kompilere og installere hver pakke. Fordi Linux Fra Scratch kan bygges på mange forskjellige systemer, er det umulig å gi nøyaktige tidsanslag. Den største pakken (Glibc) vil ta omtrent 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 kompileres fra denne boken er binutils i Kapittel 5. Tiden det tar å kompilere med én 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 10 minutter å kompilere og installer det første passet med binutils, vil det ta *omtrent* 45 minutter å bygge denne eksempelpakken. Heldigvis er de fleste byggetidene kortere enn 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.

Note

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

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

eller bare bygge med:

```
make -j4
```

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

Tidene som presenteres her 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 idé fordi den kan gi en “tilregnelighetssjekk ” som indikerer at alt er kompilert 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.

Note

Å kjører testpakkene i Kapittel 5 og Kapittel 6 er meningsløst; siden testprogrammene er kompilert 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/11.3/* for å bekrefte om disse feilene er forventet eller ikke. Denne siden er gyldig for alle tester i denne boken.

Part III. Bygge LFS Kryssverktøykjede og midlertidige verktøy

Viktig foreløpig materiale

Introduksjon

Denne delen er delt inn i tre stadier: først bygge en krysskompilator og tilhørende biblioteker; for det andre, bruke denne kryssverktøykjeden 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.

Important

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

Note

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

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

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

bygg
: er maskinen der vi bygger programmer. Merk at denne maskinen 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 de kjører på. De andre kompilatorene ccA og ccC produserer kode for maskinen de kjører på. Slike kompilatorer kalles *lokale* kompilatorer.

Implementering av Krysskompilering for LFS

Note

Alle de krysskompilerte 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 kjørbar 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 `shlib-versions` filen i roten til Glibc kildetreet.

For å forfalske en krysskompilering i LFS, navnet på vertstripletten justeres litt ved å endre "vendor" feltet i `LFS_TGT` 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 cc1 ved å bruke cc-pc på pc
2	pc	lfs	lfs	Bygg kompilator cc-lfs ved

Stadie	Bygg	Vert	Mål	Handling
				å bruke cc1 på pc
3	lfs	lfs	lfs	Bygge om og teste cc-lfs ved å bruke seg selv på lfs

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

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 triplettene for pc og lfs er kompatible med hverandre, en kjørbar fil for lfs må avhenge av glibc-2.37; 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 reinstallerte 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** kjøringer av både GCC og Glibc utfører forskjellige funksjonstester på assembleren og lenker for å bestemme hvilke programvarefunksjoner som skal aktiveres eller deaktiveres. Dette er viktigere enn man kanskje først er klar over. En feilkonfigurert GCC eller Glibc kan resultere i en subtilt ødelagt verktøykjede, hvor virkningen av et slikt brudd ikke vises før mot slutten av konstruksjonen av hele distribusjonen. En feil i testserien vil vanligvis fremheve denne feilen før det utføres for mye tilleggsarbeid.

Binutils installerer sin assembler og lenker på to steder, `$LFS/tools/bin` og `$LFS/tools/$LFS_TGT/bin`. Verktøyene i en plassering er hardlenket til den andre. En viktig fasett av lenkeren er bibliotekets søkerekkefølge. Detaljert informasjon kan fås fra **ld** ved å gi den *`--verbose`* flagget. For eksempel, **$LFS_TGT-ld --verbose | grep SEARCH** vil illustrere gjeldende søkestier og rekkefølgen deres. (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 konfigureringsskript 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 er generelt ikke et problem siden glibc vil alltid bruke kompilatoren som er relatert til *`--host`* parameteret sendt til konfigureringsskriptet; f.eks. i vårt tilfelle kompilatoren vil være **$LFS_TGT-gcc**. De binære verktøyene og kjerne deklarasjoner kan være litt mer kompliserte. Derfor tar vi ingen risiko og bruker de tilgjengelige konfigurasjonsbryterne for å fremtvinge de riktige valgene. Etter kjøring av **configure**, sjekk innholdet i `config.make` filen i `build` mappen for alle viktige detaljer. Legg merke til bruken av *`CC="$LFS_TGT-gcc"`* (med `$LFS_TGT` utvidet) for å kontrollere hvilke binære verktøy som brukes og bruken av *`-nostdinc`* og *`-isystem`* flagg for å kontrollere kompilatorens inkluderte søkeveier. 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 konfigureringsskript, `CC_FOR_TARGET` ender opp som **cc** når verten er den samme som målet, men er forskjellig fra byggesystemet. Det er derfor *`CC_FOR_TARGET=$LFS_TGT-gcc`* er 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

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

- Flere pakker oppdateres før kompilering, men bare når 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 én gang, er ikke lappen 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:

Important

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.

Important

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 kildekodetre et annet sted kan ødelegge lenker og 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.

Chapter 5. Kompilere en kryssverktøykjede

5.1. Introduksjon

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

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

5.2. Binutils-2.40 - Pass 1

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

Omtrentlig byggetid: 1 SBU
Nødvendig diskplass: 639 MB

5.2.1. Installasjon av Kryss Binutils

Note

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
```

Note

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

Forbered nå Binutils til kompilering:

```
../configure --prefix=$LFS/tools \
             --with-sysroot=$LFS \
             --target=$LFS_TGT   \
             --disable-nls       \
             --enable-gprofng=no \
             --disable-werror
```

Betydningen av konfigurasjonsalternativene:

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

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

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

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

`--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.

Fortsett med å kompilere pakken:

```
make
```

Installer pakken:

```
make install
```

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

5.3. GCC-12.2.0 - Pass 1

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

Omtrentlig byggetid: 3.3 SBU
Nødvendig diskplass: 3.8 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:

Note

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-12.2.0 tarballen fra kildemappen , og endre deretter til den opprettede mappen. Først da bør du fortsett med instruksjonene nedenfor.

```
tar -xf ../mpfr-4.2.0.tar.xz
mv -v mpfr-4.2.0 mpfr
tar -xf ../gmp-6.2.1.tar.xz
mv -v gmp-6.2.1 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=2.37 \
    --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=2.37`

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 tverrkompilator, 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 krysskompilator og er ikke nødvendig for oppgaven med å krysskompilere den midlertidige libc.

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

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

Kompiler GCC ved å kjøre:

```
make
```

Installer pakken:

```
make install
```

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:

Note

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)`/install-tools/include/limits.h
```

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

5.4. Linux-6.1.11 API Deklarasjoner

Linux API deklarasjonene (i linux-6.1.11.tar.xz) eksponerer kjernens API for bruk av Glibc.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 1.5 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-2.37

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: 1.5 SBU
Nødvendig diskplass: 822 MB

5.5.1. Installasjon av Glibc

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

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

Note

Kommandoen ovenfor er riktig. **ln** kommandoen har flere syntaktiske versjoner, så sørg for å sjekke **info coreutils ln** og `ln(1)` før du rapporterer det 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 kompatible steder:

```
patch -Np1 -i ../glibc-2.37-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=3.2                \
      --with-headers=$LFS/usr/include    \
      libc_cv_slibdir=/usr/lib
```

Betydningen av konfigureringsalternativene:

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

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

`--enable-kernel=3.2`

Dette forteller Glibc å kompilere biblioteket med støtte for 3.2 og senere Linux kjerner. Løsninger for eldre kjerner er ikke aktivert.

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

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

`libc_cv_slibdir=/usr/lib`

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

I løpet av dette stadiet kan følgende advarsel vises:

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

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

Note

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

Kompiler pakken:

```
make
```

Installer pakken:

Warning

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

Caution

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

```
echo 'int main(){}' | $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 korriger det. Dette problemet må løses før fortsetter.

Når alt er bra, rydd opp i testfilene:

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

Note

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.

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

```
$LFS/tools/libexec/gcc/$LFS_TGT/12.2.0/install-tools/mkheaders
```

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

5.6. Libstdc++ fra GCC-12.2.0

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: 1.1 GB

5.6.1. Installasjon av Target Libstdc++

Note

Libstdc++ er en del av GCC kildene. Du bør først pakke ut GCC tarball og bytte til `gcc-12.2.0` 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++/12.2.0
```

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/$LFS_TGT/include/c++/12.2.0`
Dette spesifiserer installasjonsmappen for include filer. Fordi libstdc++ er standard C++ biblioteket for LFS, skal denne mappen samsvare med plasseringen der C++ kompilatoren (**$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++/12.2.0`. 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++,stdc++fs,supc++}.la
```

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

Chapter 6. Krysskompilering av midlertidige verktøy

6.1. Introduksjon

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

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

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

6.2. M4-1.4.19

M4 pakken inneholder en makroprosessor.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 31 MB

6.2.1. Installasjon av M4

Forbered M4 for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

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

6.3. Ncurses-6.4

Ncurses pakken inneholder biblioteker for terminaluavhengig håndtering av karakterskjermer.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 51 MB

6.3.1. Installasjon av Ncurses

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

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

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

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

Forbered Ncurses for kompilering:

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

Betydningen av de nye konfigureringsalternativene:

`--with-manpage-format=normal`
Dette forhindrer Ncurses fra å installere 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 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.

`--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.

`--enable-widec`
Denne bryteren gjør at biblioteker med brede tegn (wide-character) f.eks., `libncursesw.so.6.4`) skal bygges i stedet for vanlige (f.eks., `libncurses.so.6.4`). Disse brede tegnbibliotekene er brukbare i både multibyte og tradisjonelle 8-biters lokaliteter, mens vanlige biblioteker fungerer som de skal bare i 8-biters lokaliteter. Brede karakterer og normale biblioteker er kildekompatibel, men ikke binærkompatibel.

Kompiler pakken:

```
make
```

Installer pakken:

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

Betydningen av installasjonsalternativene:

`TIC_PATH=$(pwd)/build/progs/tic`
Vi må sende stien til den nettopp bygde **tic** programmet som kjører på byggemaskinen, slik at terminaldatabasen kan opprettes uten feil.

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

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

6.4. Bash-5.2.15

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

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 67 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 Section 8.34.2, “Innholdet i Bash.”

6.5. Coreutils-9.1

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

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 162 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 Section 8.54.2, “Innhold i Coreutils.”

6.6. Diffutils-3.9

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

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 26 MB

6.6.1. Installasjon av Diffutils

Forbered Diffutils for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

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

6.7. File-5.44

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

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 36 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 Section 8.10.2, “Innholdet i File.”

6.8. Findutils-4.9.0

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: 42 MB

6.8.1. Installasjon av Findutils

Forbered Findutils for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.58.2, “Innhold i Findutils.”

6.9. Gawk-5.2.1

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 Section 8.57.2, “Innhold i Gawk.”

6.10. Grep-3.8

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

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 25 MB

6.10.1. Installasjon av Grep

Forbered Grep for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

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

6.11. Gzip-1.12

Gzip pakken inneholder programmer for komprimering og dekomprimering av filer.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 11 MB

6.11.1. Installasjon av Gzip

Forbered Gzip for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

Detaljer om denne pakken finner du i Section 8.61.2, “Contents of Gzip.”

6.12. Make-4.4

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

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 15 MB

6.12.1. Installasjon av Make

Først, fiks et problem identifisert oppstrøms:

```
sed -e '/ifdef SIGPIPE/,+2 d' \
    -e '/undef  FATAL_SIG/i FATAL_SIG (SIGPIPE);' \
    -i src/main.c
```

Forbered Make for kompilering:

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

Betydningen av det nye konfigureringsalternativet:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

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

6.13. Patch-2.7.6

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

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 12 MB

6.13.1. Installasjon av Patch

Forbered Patch for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

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

6.14. Sed-4.9

Sed pakken inneholder en dataflyt (stream) redigerer.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 19 MB

6.14.1. Installasjon av Sed

Forbered Sed for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

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

6.15. Tar-1.34

Tar pakken gir muligheten til å lage tar arkiver 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: 38 MB

6.15.1. Installasjon av Tar

Forbered Tar for kompilering:

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

Kompiler pakken:

```
make
```

Installer pakken:

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

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

6.16. Xz-5.4.1

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: 20 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-5.4.1
```

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 Section 8.8.2, “Innhold i Xz.”

6.17. Binutils-2.40 - Pass 2

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

Omtrentlig byggetid: 0.4 SBU
Nødvendig diskplass: 525 MB

6.17.1. Installation of Binutils

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

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

Opprett en egen byggemappe igjen:

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

Forbered Binutils for kompilering:

```
../configure                   \
    --prefix=/usr              \
    --build=$(../config.guess) \
    --host=$LFS_TGT            \
    --disable-nls              \
    --enable-shared            \
    --enable-gprofng=no        \
    --disable-werror           \
    --enable-64-bit-bfd
```

Betydningen av de nye konfigureringsalternativene:

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

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

Kompiler pakken:

```
make
```

Installer pakken:

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

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}.{a,la}
```

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

6.18. GCC-12.2.0 - Pass 2

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

Omtrentlig byggetid: 4.6 SBU
Nødvendig diskplass: 4.7 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-4.2.0.tar.xz
mv -v mpfr-4.2.0 mpfr
tar -xf ../gmp-6.2.1.tar.xz
mv -v gmp-6.2.1 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 byggeregelen 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 overstyrer 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-libgomp                              \
    --disable-libquadmath                          \
    --disable-libssp                               \
    --disable-libvtv                               \
    --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 dennne kompilatoren at den må lete etter overskrifter og biblioteker i `$LFS`. Men byggesystemet til GCC bruker andre verktøy som ikke er klar over denne plasseringen. Denne bryteren sørger for at de finner de nødvendige filene på `$LFS`, og ikke på verten.

`--target=$LFS_TGT`

Vi krysskompilerer GCC, så det er umulig å bygge målbibliotekene (`libgcc` og `libstdc++`) med tidligere kompilerte GCC binærfiler—disse binærfilene vil ikke kjøre på verten. GCC byggesystemet vil forsøke å bruke vertens 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 den delte `libgcc` som ble bygget i dette passet, i stedet for den statiske versjonen bygget i GCC pass 1. Dette er nødvendig for å støtte C++ unntakshåndtering.

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 Section 8.26.2, “Innhold i GCC.”

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

7.1. Introduksjon

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

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

Inntil Section 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

Note

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

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

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

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

7.3. Forberede det virtuelle kjernefilsystemer

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 Section 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 verts-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 kjernefilsystemer

Monter nå de gjenværende virtuelle kjernefilsystemene:

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

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

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
  mkdir -pv $LFS/$(readlink $LFS/dev/shm)
else
  mount -t 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     \
    /bin/bash --login
```

`-i` alternativet gitt til **env** kommandoen vil slette alle variabler i chroot miljøet. Etter det, bare `HOME`, `TERM`, `PS1`, og `PATH` variablene settes på nytt. *`TERM=$TERM`* konstruksjonen 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.

Note

Det er viktig at alle kommandoene gjennom resten av dette kapittel og de følgende kapitlene kjøres fra chroot miljøet. Hvis du forlater dette miljøet av en eller annen grunn (omstart for eksempel), sørg for at de virtuelle kjernefilsystemene er montert som forklart i Section 7.3.1, “Montering og fylling av /dev” og Section 7.3.2, “Montering av det virtuelle kjernefilsystemer” og gå inn i chroot igjen før du fortsetter med installasjonen.

7.5. Opprette mapper

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

Note

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/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.

Warning

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

```
ln -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
uuidd:x:80:80:UUID Generation Daemon User:/dev/null:/usr/bin/false
nobody:x:65534:65534:Unprivileged User:/dev/null:/usr/bin/false
EOF
```

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:
usb:x:14:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
input:x:24:
mail:x:34:
kvm:x:61:
uuidd:x:80:
wheel:x:97:
users:x:999:
nogroup:x:65534:
EOF
```

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 gruppenavnoppløsning 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 påloggінger og utloggінger. `/var/log/lastlog` filen registrerer når hver bruker sist logget på. `/var/log/faillog` filen registrerer mislykkede påloggingsforsøk. `/var/log/btmp` filen registrerer de dårlige påloggingsforsøkene.

Note

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

7.7. Gettext-0.21.1

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: 1.0 SBU
Nødvendig diskplass: 287 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 Section 8.31.2, “Innhold i Gettext.”

7.8. Bison-3.8.2

Bison pakken inneholder en parsergenerator.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 57 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 Section 8.32.2, “Innholdet i Bison.”

7.9. Perl-5.36.0

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

Omtrentlig byggetid: 0.6 SBU
Nødvendig diskplass: 281 MB

7.9.1. Installasjon av Perl

Forbered Perl for kompilering:

```
sh Configure -des                                         \
             -Dprefix=/usr                                \
             -Dvendorprefix=/usr                          \
             -Dprivlib=/usr/lib/perl5/5.36/core_perl      \
             -Darchlib=/usr/lib/perl5/5.36/core_perl      \
             -Dsitelib=/usr/lib/perl5/5.36/site_perl      \
             -Dsitearch=/usr/lib/perl5/5.36/site_perl     \
             -Dvendorlib=/usr/lib/perl5/5.36/vendor_perl \
             -Dvendorarch=/usr/lib/perl5/5.36/vendor_perl
```

Betydningen av de nye konfigureringsalternativene:

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

Kompiler pakken:

```
make
```

Installer pakken:

```
make install
```

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

7.10. Python-3.11.2

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: 0.4 SBU
Nødvendig diskplass: 529 MB

7.10.1. Installasjon av Python

Note

Det er to pakkefiler som navnet begynner med “python”. Den å pakke ut er `Python-3.11.2.tar.xz` (legg merke til stor bokstav først).

Forbered Python for kompilering:

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

Betydningen av konfigureringsalternativet:

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

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

Kompiler pakken:

```
make
```

Note

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

Installer pakken:

```
make install
```

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

7.11. Texinfo-7.0.2

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

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 116 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 Section 8.68.2, “Innhold i Texinfo.”

7.12. Util-linux-2.38.1

Util-linux pakken inneholder diverse hjelpeprogrammer.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 150 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 ADJTIME_PATH=/var/lib/hwclock/adjtime    \
            --libdir=/usr/lib    \
            --docdir=/usr/share/doc/util-linux-2.38.1 \
            --disable-chfn-chsh  \
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
            --disable-runuser    \
            --disable-pylibmount \
            --disable-static     \
            --without-python     \
            runstatedir=/run
```

Betydningen av konfigureringsalternativene:

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

`--libdir=/usr/lib`
Denne bryteren sikrer 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 **uuidd** og `libuuid` riktig.

Kompiler pakken:

```
make
```

Installer pakken:

```
make install
```

Detaljer om denne pakken finner du i Section 8.73.2, “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 gjøre om noe som har blitt bygget vellykket, det og lage en sikkerhetskopi av det nåværende LFS systemet kan vise seg å være nyttig.

Note

Alle de resterende trinnene i denne delen er valgfrie. Likevel, så snart du begynner å installere pakker i 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
```

Important

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 Section 2.6, “Stille inn $LFS variabelen”.

Før 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:

Note

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-11.3.tar.xz .
```

Note

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

7.13.3. Gjenopprett

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

Warning

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

```
cd $LFS
rm -rf ./*
tar -xpf $HOME/lfs-temp-tools-11.3.tar.xz
```

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

Important

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

Part IV. Bygge LFS systemet

Chapter 8. Installere grunnleggende systemprogramvare

8.1. Introduksjon

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

Installasjonen av denne programvaren er enkel. Skjønt i mange tilfeller kan installasjonsinstruksjonene gjøres kortere og mer generelle, vi har valgt å gi de fullstendige instruksjonene for hver pakke for å minimere mulighetene for feil. Nøkkelen til å lære hva som gjør 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 installasjonsinstruksene , er det en liste over programmer og biblioteker (sammen med korte beskrivelser) som pakken installerer.

Note

SBU verdiene og nødvendig diskplass inkluderer testpakkedata for alle gjeldende pakker i 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 kapittelet fjerner eller deaktiverer vi installasjon av de fleste statiske biblioteker. Vanligvis gjøres dette ved å utstede en `--disable-static` alternativ til **configure**. I andre tilfeller er det nødvendig med alternative midler. I noen få tilfeller, spesielt glibc og gcc, forblir bruken av statiske biblioteker avgjørende for 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 inkludere:

- 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 tilfredsstiller alle målgrupper er vanskelig.

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

8.2.1. Oppgraderingsproblemer

En Pakkehåndterer gjør det enkelt å oppgradere til nyere versjoner når de 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 (og bør ikke bli, se neste element) oppgraderes ved siden av kjernen. Du må starte systemet på nytt for å bruke den oppgraderte kjernen.
- Hvis Linux API-deklarasjoner eller Glibc må oppgraderes til en nyere versjon, (f.eks. fra glibc-2.31 til glibc-2.32), er det tryggere å gjenoppbygge LFS. Selv om du *kanskje* kan gjenoppbygge alle pakkene i deres avhengighetsrekkefølge, anbefaler vi ikke det.
- 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 biblioteker med mindre alle de avhengige pakkene er rekompilert.
- Hvis en pakke som inneholder et delt bibliotek oppdateres, og navnet på biblioteket ikke endres, men versjonsnummeret til 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  -e 'libfoo.*deleted' /proc/*/maps |
   tr -cd 0-9\\n | xargs -r ps u
```

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

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

8.2.2. Pakkehåndteringsteknikker

Følgende er noen vanlige pakkehåndteringsteknikker. Før du ta en avgjørelse om en pakkeforvalter, gjør litt 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 `/usr/pkg/foo-1.1` og en symbolkobling er laget fra `/usr/pkg/foo` til `/usr/pkg/foo-1.1`. Når en ny versjon foo-1.2 kommer, blir den installert i `/usr/pkg/foo-1.2` og den forrige symbolkoblingen erstattes av en symbolkobling til den nye versjonen.

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

8.2.2.3. Symlink Style Package Management

Dette er en variant av den tidligere pakkehåndteringsteknikken. Hver pakke er installert 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. Dette tilnærmingen fungerer som følger:

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

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

8.2.2.4. Tidsstempelbasert

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

Selv om denne ordningen har fordelen av å være enkel, har den to ulemper. Hvis filene under installasjonen er installert med et annet tidsstempel enn gjeldende tid, vil disse filene ikke spores av pakkebehandleren. Dessuten kan denne ordningen bare brukes når én pakke installeres om gangen. Loggene er ikke pålitelige hvis to pakker installeres på to forskjellige konsoller.

8.2.2.5. Sporing av installasjonsskript

I denne tilnærmingen, blir kommandoene som installasjonsskriptene utfører registrert. Det er to teknikker man kan bruke:

`LD_PRELOAD` miljøvariabelen kan settes til å peke på et bibliotek som skal forhåndslastes før installasjonen. Under installasjonen, sporer dette biblioteket pakkene som blir installert og fester seg til ulike kjørbare filer som f.eks **cp**, **install**, **mv** 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 installasjonensskriptet.

8.2.2.6. Opprette pakkearkiver

I denne ordningen er pakkeinstallasjonen forfalsket til et separat tre som beskrevet i Symlink pakkebehandlingenen. Etter installasjon, opprettes et pakkearkiv ved hjelp av de installerte filene. Dette arkivet brukes deretter til å installere pakken enten på den lokale maskin eller kan til og med brukes til å installere pakken på andre maskiner.

Denne tilnærmingen brukes av de fleste pakkebehandlere som finnes i kommersielle distribusjoner. Eksempler på pakkeforvaltere som følger dette tilnærmingen er RPM (som for øvrig kreves av *Linux Standard Base Specification*), pkg-utils, Debian's apt, og Gentoo's Portage system. Et hint som beskriver hvordan du adopterer denne stilen av pakkehåndtering for LFS systemer ligger på *https://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt*.

Oppretting av pakkefiler som inkluderer avhengighetsinformasjon er kompleks og er utenfor omfanget av LFS.

Slackware bruker et **tar** basert system for pakke arkiv. Dette systemet håndterer med vilje ikke pakkeavhengigheter som mer komplekse pakkeforvaltere gjør. For detaljer om Slackware pakkebehandling, se *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.

Note

Det har vært noen rapporter om problemer ved kopiering mellom lignende, men ikke identiske arkitekturer. For eksempel instruksjonssettet for et Intel-system er ikke identisk med en AMD-prosessor og nyere versjoner av enkelte prosessorer kan ha instruksjoner som ikke er tilgjengelige i tidligere versjoner.

Til slutt må det nye systemet gjøres oppstartbart via Section 10.4, “Bruke GRUB til å sette opp oppstartsprosessen”.

8.3. Man-pages-6.03

Man-pages pakken inneholder over 2400 mansider..

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 32 MB

8.3.1. Installasjon av Man-pages

Installer Man-pages ved å kjøre:

```
make prefix=/usr install
```

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-20230202

Iana-Etc pakken leverer data for nettverkstjenester og protokoller.

Omtrentlig byggetid: mindre enn 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

`/etc/protocols` Beskriver de ulike DARPA Internettprotokollene som er tilgjengelig fra TCP/IP undersystemet

`/etc/services` Gir en tilordning mellom vennlige tekstnavn for internetttjenester, og deres underliggende tildelte portnumre og protokolltyper

8.5. Glibc-2.37

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: 11 SBU
Nødvendig diskplass: 2.9 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-2.37-fhs-1.patch
```

Fiks et sikkerhetsproblem identifisert oppstrøms:

```
sed '/width -=/s/workend - string/number_length/' \
    -i stdio-common/vfprintf-process-arg.c
```

Glibc dokumentasjonen anbefaler å bygge Glibc i en dedikert byggemappe:

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

Sørg for at **ldconfig** og **sln** verktøyene vil bli installert i `/usr/sbin`:

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

Forbered Glibc for kompilering:

```
../configure --prefix=/usr                            \
             --disable-werror                         \
             --enable-kernel=3.2                      \
             --enable-stack-protector=strong          \
             --with-headers=/usr/include              \
             libc_cv_slibdir=/usr/lib
```

Betydningen av konfigureringsalternativene:

`--disable-werror`
Dette alternativet deaktiverer alternativet -Werror sendt til GCC. Dette er nødvendig for å kjøre testpakken.

`--enable-kernel=3.2`
Dette alternativet forteller byggesystemet at denne glibc kan brukes med kjerner så gamle som 3.2. Dette betyr å generere løsninger i tilfelle et systemanrop introdusert i en senere versjon ikke kan brukes.

`--enable-stack-protector=strong`
Dette alternativet øker systemsikkerheten ved å legge til ekstra kode for å se etter bufferoverflyt "buffer overflows", for eksempel stabel (stack) knusende angrep.

`--with-headers=/usr/include`
Dette alternativet forteller byggesystemet hvor det skal finne kjernens API deklarasjoner.

`libc_cv_slibdir=/usr/lib`
Denne variabelen setter riktig bibliotek for alle systemer. Vi ønsker ikke at lib64 skal brukes.

Kompiler pakken:

```
make
```

Important

I denne delen anses testpakken for Glibc som kritisk. Ikke hopp over den under noen omstendigheter.

Vanligvis består ikke noen få tester. Testfeilene som er oppført nedenfor er vanligvis trygge å ignorere.

```
make check
```

Du kan se noen testfeil. Glibc testpakken er noe avhengig av vertssystemet. Noen få feil ut av over 5000 tester kan generelt ignoreres. Dette er en liste over de vanligste problemene som er sett for nyere versjoner av LFS:

- *io/tst-lchmod* er kjent for å mislykkes i LFS chroot miljøet.
- *misc/tst-ttyname* er kjent for å mislykkes i LFS chroot miljøet.
- *stdlib/tst-arc4random-thread* testen er kjent for å mislykkes hvis vertskjernen er relativt gammel.
- Noen tester, for eksempel *nss/tst-nss-files-hosts-multi*, er kjent for å mislykkes på relativt trege systemer på grunn av et internt tidsavbrudd.

Selv om det er en ufarlig melding, vil installasjonsstadiet til Glibc klage på fravær av `/etc/ld.so.conf`. Forhindre denne advarselen med:

```
touch /etc/ld.so.conf
```

Fiks Makefilen til å hoppe over en unødvendig sunnhetssjekk som svikter i LFS delmiljøet:

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

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 konfigurasjonsfilen og kjøretidsmappen for **nscd**:

```
cp -v ../nscd/nscd.conf /etc/nscd.conf
mkdir -pv /var/cache/nscd
```

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:

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

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

Alternativt kan du installere alle lokaliteter som er oppført i `glibc-2.37/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-2.37/localedata/SUPPORTED` filen når du trenger dem. For eksempel er følgende to lokaliteter nødvendig for noen tester senere i dette kapittelet:

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

Note

Glibc bruker nå libidn2 når den løser internasjonalisert domenenavn. Dette er en kjøretids avhengighet. Hvis denne evnen er nødvendig, er instruksjonene for installasjon av libidn2 i *BLFS libidn2 siden*.

8.5.2. 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 ../../tzdata2022g.tar.gz

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

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

cp -v zone.tab zone1970.tab iso3166.tab $ZONEINFO
zic -d $ZONEINFO -p America/New_York
unset ZONEINFO
```

Betydningen av zic kommandoene:

`zic -L /dev/null ...`

Dette skaper posix tidssoner uten noen skuddsekunder. Det er konvensjonelt å legge disse i både `zoneinfo` og `zoneinfo/posix`. Det er nødvendig for å legge POSIX tidssonene i `zoneinfo`, ellers vil forskjellige testpakker rapportere feil. På et innebygd system, hvor plass er stramt og du aldri har tenkt å oppdatere tidssonene, kan du spare 1,9 MB ved å ikke bruke `posix` mappen, men noen applikasjoner eller testpakker kan produsere noen feil.

`zic -L leapseconds ...`

Dette skaper riktige tidssoner, inkludert skuddsekunder. På en innebygd system, hvor det er trangt om plass og du ikke har tenkt å oppdatere tidssonene noen gang, eller ikke bryr deg om riktig tid, kan du spar 1,9 MB ved å utelate `right` mappen.

`zic ... -p ...`

Dette oppretter `posixrules` filen. Vi bruker New York fordi POSIX krever at reglene for sommertid er i samsvar med amerikanske regler.

En måte å bestemme den lokale tidssonen på er å kjøre følgende skript:

```
tzselect
```

Etter å ha svart på noen spørsmål om lokaliteten, vil skriptet skrive ut navnet på tidssonen (f.eks., *America/Edmonton*). Det er også noen andre mulige tidssoner oppført i `/usr/share/zoneinfo` som for eksempel *Canada/Eastern* eller *EST5EDT* som ikke identifiseres av skriptet, men kan brukes.

Deretter oppretter du `/etc/localtime` filen med å kjøre:

```
ln -sfv /usr/share/zoneinfo/<xxx> /etc/localtime
```

Erstatt *<xxx>* med navnet på den valgte tidssonen (f.eks. 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, nscd, pcprofiledump, pldd, sln, sotruss, sprof, tzselect, xtrace, zdump, og zic

Installerte biblioteker: ld-linux-x86-64.so.2, ld-linux.so.2, libBrokenLocale.{a,so}, libanl.{a,so}, libc.{a,so}, libc_nonshared.a, libc_malloc_debug.so, libcrypt.{a,so}, libdl.{a,so.2}, libg.a, libm.{a,so}, libmcheck.a, libmemusage.so, libmvec.{a,so}, libnsl.so.1, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libpcprofile.so, libpthread.{a,so.0}, libresolv.{a,so}, librt.{a,so.1}, libthread_db.so, og libutil.{a,so.1}

Installerte mapper: /usr/include/arpa, /usr/include/bits, /usr/include/gnu, /usr/include/net, /usr/include/netash, /usr/include/netatalk, /usr/include/netax25, /usr/include/neteconet, /usr/include/netinet, /usr/include/netipx, /usr/include/netiucv, /usr/include/netpacket, /usr/include/netrom, /usr/include/netrose, /usr/include/nfs, /usr/include/protocols, /usr/include/rpc, /usr/include/sys, /usr/lib/audit, /usr/lib/gconv, /usr/lib/locale, /usr/libexec/getconf, /usr/share/i18n, /usr/share/zoneinfo, /var/cache/nscd, og /var/lib/nss_db

Korte beskrivelser

gencat	Genererer meldingskataloger
getconf	Viser systemkonfigurasjonsverdiene for filsystem spesifikke variabler
getent	Henter oppføringer fra en administrativ database
iconv	Utfører tegnsettkonvertering
iconvconfig	Skaper hurtiglastings **iconv** modulkonfigurasjons filer
ldconfig	Konfigurerer dynamiske lenker til kjøretidsbindingene
ldd	Rapporter hvilke delte biblioteker som kreves av hvert gitt program eller delte bibliotek
lddlibc4	Assisterer **ldd** med objektfiler. Det eksisterer ikke på nyere arkitekturer som x86_64
locale	Skriver ut forskjellig informasjon om gjeldende lokalitet
localedef	Kompilerer lokalitetsspesifikasjoner
makedb	Oppretter en enkel database fra tekst inndata
mtrace	Leser og tolker en minnesporingsfil og viser et sammendrag i menneskelestbart format
nscd	En nisse (daemon) som gir et hurtiglager for de vanligste navnetjenesteforespørsler
pcprofiledump	Dumper informasjon generert av PC profiling
pldd	Viser dynamiske delte objekter som brukes av kjørende prosesser
sln	En statisk koblet **ln** program
sotruss	Sporer delte biblioteksprosedyrekall for en spesifisert kommando
sprof	Leser og viser profileringsdata for delte objekter
tzselect	Spør brukeren om lokaliteten til systemet og rapporterer den tilsvarende tidssonebeskrivelsen
xtrace	Sporer kjøringen av et program ved å skrive ut gjeldende utført funksjon
zdump	Tidssone dumperen
zic	Tidssonekompilatoren

`ld-*.so`	Hjelpeprogrammet for kjørbare delte biblioteker
`libBrokenLocale`	Brukes internt av Glibc som et grovt hack for å få ødelagte programmer (f.eks. noen Motif-applikasjoner) kjørende. Se kommentarer i `glibc-2.37/locale/broken_cur_max.c` for mer informasjon
`libanl`	Et asynkront navneoppslagsbibliotek
`libc`	C hovedbiblioteket
`libc_malloc_debug`	Slår på minneallokeringskontroll når den er forhåndslastet
`libcrypt`	Kryptografibiblioteket
`libdl`	Dummy bibliotek som ikke inneholder noen funksjoner. Tidligere var den dynamisk koblingsgrensesnittbibliotek, funksjonene er nå i `libc`
`libg`	Dummy bibliotek som ikke inneholder noen funksjoner. Tidligere var det et kjøretidsbibliotek for **g++**
`libm`	Det matematiske biblioteket
`libmvec`	Matematisk vektorbibliotek, koblet inn etter behov når `libm` blir brukt
`libmcheck`	Slår på minneallokeringskontroll når den er koblet til
`libmemusage`	Brukt av **memusage** for å hjelpe til med å samle inn informasjon om minnebruken til et program
`libnsl`	Nettverkstjenestebiblioteket, nå avviklet
`libnss_*`	Navnetjenestebrytermodulene, som inneholder funksjoner for å løse vertsnavn, brukernavn, gruppenavn, aliaser, tjenester, protokoller osv. Lastet av `libc` ifølge konfigurasjon i `/etc/nsswitch.conf`
`libpcprofile`	Kan forhåndslastes til PC profile en kjørbar fil
`libpthread`	Dummy bibliotek som ikke inneholder noen funksjoner. Tidligere inneholdt den funksjoner som gir de fleste grensesnittene som er spesifisert av POSIX.1b Realtime Extension, nå er funksjonene i `libc`
`libresolv`	Inneholder funksjoner for å lage, sende og tolke pakker til domenenavnservere
`librt`	Inneholder funksjoner som gir de fleste grensesnittene som er spesifisert av POSIX.1b Realtime Extension
`libthread_db`	Inneholder funksjoner som er nyttige for å bygge feilsøkere for flertrådede programmer
`libutil`	Dummy bibliotek som ikke inneholder noen funksjoner. Tidligere inneholdt den kode for “standard” funksjoner som brukes i mange forskjellige Unix verktøy. Disse funksjonene er nå i `libc`

8.6. Zlib-1.2.13

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

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 6.2 MB

8.6.1. Installasjon av Zlib

Forbered Zlib for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

Fjern et ubrukelig statisk bibliotek:

```
rm -fv /usr/lib/libz.a
```

8.6.2. Innhold i Zlib

Installerte biblioteker: libz.so

Korte beskrivelser

`libz` Inneholder komprimerings- og dekompresjonsfunksjoner som brukes av noen programmer

8.7. Bzip2-1.0.8

Bzip2 pakken inneholder programmer for komprimering og dekomprimering av filer. Komprimering av tekstfiler med **bzip2** gir mye bedre kompresjonsprosent enn med den tradisjonelle **gzip**.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 7.2 MB

8.7.1. Installasjon av Bzip2

Bruk en oppdatering som vil installere dokumentasjonen for denne pakken:

```
patch -Np1 -i ../bzip2-1.0.8-install_docs-1.patch
```

Følgende kommando sikrer at installasjonen av symbolske lenker er relative:

```
sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
```

Sørg for at mansidene er installert på riktig sted:

```
sed -i "s@(PREFIX)/man@(PREFIX)/share/man@g" Makefile
```

Forbered Bzip2 for kompilering med:

```
make -f Makefile-libbz2_so
make clean
```

Betydningen av make parameteren:

`-f Makefile-libbz2_so`

Vil føre til at Bzip2 bygges med en annen `Makefile` fil, i dette tilfellet `Makefile-libbz2_so` filen, som skaper en dynamisk `libbz2.so` bibliotek og lenker Bzip2 verktøyene mot det.

Kompiler og test pakken:

```
make
```

Installer programmene:

```
make PREFIX=/usr install
```

Installer det delte biblioteket:

```
cp -av libbz2.so.* /usr/lib
ln -sv libbz2.so.1.0.8 /usr/lib/libbz2.so
```

Installer den delte **bzip2** binær inn i `/usr/bin` mappen, og erstatt to eksemplarer av **bzip2** med symbolske lenker:

```
cp -v bzip2-shared /usr/bin/bzip2
for i in /usr/bin/{bzcat,bunzip2}; do
  ln -sfv bzip2 $i
done
```

Fjern et ubrukelig statisk bibliotek:

```
rm -fv /usr/lib/libbz2.a
```

8.7.2. Innhold i Bzip2

Installerte programmer: bunzip2 (linker til bzip2), bzcat (linker til bzip2), bzcmp (linker til bzdiff), bzdiff, bzegrep (linker til bzgrep), bzfgrep (linker til bzgrep), bzgrep, bzip2, bzip2recover, bzless (linker til bzmore), og bzmore
Installerte biblioteker: libbz2.so
Installert mappe: /usr/share/doc/bzip2-1.0.8

Korte beskrivelser

bunzip2 Dekomprimerer bzippede filer

bzcat Dekomprimerer til standard utgang

bzcmp Kjører **cmp** på bzippede filer

bzdiff Kjører **diff** på bzippede filer

bzegrep Kjører **egrep** på bzippede filer

bzfgrep Kjører **fgrep** på bzippede filer

bzgrep Kjører **grep** på bzippede filer

bzip2 Komprimerer filer ved å bruke Burrows-Wheeler sortering på blokktekst komprimeringsalgoritme med Huffman-koding; kompresjonshastigheten er bedre enn det som oppnås med mer konvensjonelle kompressorer som bruker “Lempel-Ziv” algoritmer, som **gzip**

bzip2recover Prøver å gjenopprette data fra skadde bzippede filer

bzless Kjører **less** på bzippede filer

bzmore Kjører **more** på bzippede filer

`libbz2` Biblioteket implementerer tapsfri, blokksorterende datakomprimering ved å bruke Burrows-Wheeler-algoritmen

8.8. Xz-5.4.1

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-5.4.1
```

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.8.2. Innhold i Xz

Installerte programmer: lzcat (lenker til xz), lzcmp (lenker til xzdiff), lzdiff (lenker til xzdiff), lzegrep (lenker til xzgrep), lzfgrep (lenker til xzgrep), lzgrep (lenker til xzgrep), lzless (lenker til xzless), lzma (lenker til xz), lzmadec, lzmainfo, lzmore (lenker til xzmore), unlzma (lenker til xz), unxz (lenker til xz), xz, xzcat (lenker til xz), xzcmp (lenker til xzdiff), xzdec, xzdiff, xzegrep (lenker til xzgrep), xzfgrep (lenker til xzgrep), xzgrep, xzless, og xzmore
Installerte biblioteker: liblzma.so
Installerte mapper: /usr/include/lzma og /usr/share/doc/xz-5.4.1

Korte beskrivelser

lzcat Dekomprimerer til standard utgang
lzcmp Kjører **cmp** på LZMA komprimerte filer
lzdiff Kjører **diff** på LZMA komprimerte filer
lzegrep Kjører **egrep** på LZMA komprimerte filer
lzfgrep Kjører **fgrep** på LZMA komprimerte filer
lzgrep Kjører **grep** på LZMA komprimerte filer
lzless Kjører **less** på LZMA komprimerte filer
lzma Komprimerer eller dekomprimerer filer ved å bruke LZMA formatet
lzmadec En liten og rask dekoder for LZMA komprimerte filer
lzmainfo Viser informasjon som er lagret i den komprimerte LZMA filoverskriften

lzmore Kjører **more** på LZMA komprimerte filer

unlzma Dekomprimerer filer ved å bruke LZMA formatet

unxz Dekomprimerer filer ved å bruke XZ formatet

xz Komprimerer eller dekomprimerer filer ved å bruke XZ formatet

xzcat Dekomprimerer til standard utgang

xzcmp Kjører **cmp** på XZ komprimerte filer

xzdec En liten og rask dekoder for XZ komprimerte filer

xzdiff Kjører **diff** på XZ komprimerte filer

xzegrep Kjører **egrep** på XZ komprimerte filer

xzfgrep Kjører **fgrep** på XZ komprimerte filer

xzgrep Kjører **grep** på XZ komprimerte filer

xzless Kjører **less** på XZ komprimerte filer

xzmore Kjører **more** på XZ komprimerte filer

`liblzma` Bibliotek som implementerer tapsfri, blokksorterende data komprimering ved å bruke Lempel-Ziv-Markov kjedealgoritmen

8.9. Zstd-1.5.4

Zstandard er en sanntidskomprimeringsalgoritme som gir høyt kompresjonsforhold. Den tilbyr et veldig bredt spekter av kompresjons/hastighets avveininger, samtidig som den støttes av en veldig rask dekoder.

Omtrentlig byggetid: 0.4 SBU
Nødvendig diskplass: 75 MB

8.9.1. Installasjon av Zstd

Kompiler pakken:

```
make prefix=/usr
```

Note

I testutdataen er det flere steder som angir 'failed'. Disse er forventet og bare 'FAIL' er en reell testfeil. Det skal ikke være noen testfeil.

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make prefix=/usr install
```

Fjern det statiske biblioteket:

```
rm -v /usr/lib/libzstd.a
```

8.9.2. Innhold i Zstd

Installerte programmer: zstd, zstdcat (lenker til zstd), zstdgrep, zstdless, zstdmt (lenker til zstd), og unzstd (lenker til zstd)
Installert bibliotek: libzstd.so

Korte beskrivelser

zstd Komprimerer eller dekomprimerer filer ved å bruke ZSTD formatet

zstdgrep Kjører **grep** på ZSTD komprimerte filer

zstdless Kjører **less** på ZSTD komprimerte filer

`libzstd` Biblioteket implementerer tapsfri datakomprimering ved å bruke ZSTD algoritmen

8.10. File-5.44

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

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 16 MB

8.10.1. Installasjon av File

Forbered File for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.10.2. Innholdet i File

Installerte programmer: file
Installert bibliotek: libmagic.so

Korte beskrivelser

file Prøver å klassifisere hver gitt fil; den gjør dette ved å utføre flere tester—filsystemtester, magiske talltester og språk tester

`libmagic` Inneholder rutiner for magisk tallgjenkjenning, brukt av **file** programmet

8.11. Readline-8.2

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

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 16 MB

8.11.1. Installasjon av Readline

Å installere Readline på nytt vil føre til at de gamle bibliotekene flyttes til <libraryname>.old. Selv om dette normalt ikke er et problem, i noen tilfeller kan det utløse en koblingsfeil i **ldconfig**. Dette kan unngås ved å utstede følgende to seds:

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

Løs nå et problem identifisert oppstrøms:

```
patch -Np1 -i ../readline-8.2-upstream_fix-1.patch
```

Forbered Readline for kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --with-curses    \
            --docdir=/usr/share/doc/readline-8.2
```

Betydningen av konfigureringsalternativet:

`--with-curses`
Dette alternativet forteller Readline at det kan finne termcap bibliotekfunksjoner i curses biblioteket, i stedet for et separat termcap bibliotek. Det gjør det mulig å generere en korrekt `readline.pc` fil.

Kompiler pakken:

```
make SHLIB_LIBS="-lncursesw"
```

Betydningen av make alternativet:

`SHLIB_LIBS="-lncursesw"`
Dette alternativet tvinger Readline til å lenke mot `libncursesw` biblioteket.

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make SHLIB_LIBS="-lncursesw" install
```

Hvis ønskelig, installer dokumentasjonen:

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.2
```

8.11.2. Innhold i Readline

Installerte biblioteker: libhistory.so og libreadline.so
Installerte mapper: /usr/include/readline og /usr/share/doc/readline-8.2

Korte beskrivelser

`libhistory` Gir et konsistent brukergrensesnitt for tilbakekalling av linjer fra historien

`libreadline` Gir et sett med kommandoer for å manipulere tekst som er skrevet i en interaktiv økt av et program

8.12. M4-1.4.19

M4 pakken inneholder en makroprosessor.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 49 MB

8.12.1. Installasjon av M4

Forbered M4 for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.12.2. Innhold i M4

Installert program: m4

Korte beskrivelser

m4 Kopierer de gitte filene mens de utvider makroene som de inneholder. Disse makroene er enten innebygde eller brukerdefinerte og kan ta et hvilket som helst antall argumenter. Foruten å utføre makroutvidelse, **m4** har innebygde funksjoner for å inkludere navngitte filer, kjører Unix kommandoer, utføre heltallsaritmetikk, manipulere tekst, rekursjon osv. **m4** programmet kan brukes enten som en grenseflate til en kompilator eller som en makroprosessor på egen hånd

8.13. Bc-6.2.4

Bc pakken inneholder et vilkårlig behandlingsspråk for numerisk presisjon.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 7.6 MB

8.13.1. 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
```

8.13.2. Innholdet i Bc

Installerte programmer: bc og dc

Korte beskrivelser

bc En kommandolinjekalkulator

dc En reverse-polish kommandolinjekalkulator

8.14. 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

8.14.1. Installasjon av Flex

Forbered Flex for kompilering:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/flex-2.6.4 \
            --disable-static
```

Kompiler pakken:

```
make
```

For å teste resultatene (ca. 0,5 SBU), utsted:

```
make check
```

Installer pakken:

```
make install
```

Noen få programmer vet ikke om **flex** ennå og prøver å kjøre forgjengeren, **lex**. For å støtte de programmene, opprett en symbolsk lenke kalt `lex` som kjører `flex` i **lex** emuleringsmodus:

```
ln -sv flex /usr/bin/lex
```

8.14.2. Innhold i Flex

Installerte programmer: flex, flex++ (lenker til flex), og lex (lenker til flex)
Installerte biblioteker: libfl.so
Installert mappe: /usr/share/doc/flex-2.6.4

Korte beskrivelser

flex Et verktøy for å generere programmer som gjenkjenner mønstre i tekst; det gir mulighet for allsidigheten til å spesifisere reglene for mønstersøking, eliminere behovet for å utvikle et spesialisert program

flex++ En utvidelse av flex brukes til å generere C++ kode og klasser. Det er en symbolsk kobling til **flex**

lex En symbolsk lenke som kjører **flex** i **lex** emuleringsmodus

`libfl` `flex` biblioteket

8.15. Tcl-8.6.13

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

Omtrentlig byggetid: 2.7 SBU
Nødvendig diskplass: 89 MB

8.15.1. Installasjon av Tcl

Denne pakken og de to neste (Expect og DejaGNU) er installert for å støtte kjøring av testpakkene for binutils og GCC og andre pakker. Å installere tre pakker for testformål kan virke overdrevent, men det er veldig betryggende, om ikke avgjørende, å vite at de viktigste verktøyene fungerer som de skal.

Forbered Tcl for kompilering:

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

Bygg pakken:

```
make

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

sed -e "s|$SRCDIR/unix/pkgs/tdbc1.1.5|/usr/lib/tdbc1.1.5|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.5/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/tdbc1.1.5/library|/usr/lib/tcl8.6|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.5|/usr/include|"            \
    -i pkgs/tdbc1.1.5/tdbcConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/itcl4.2.3|/usr/lib/itcl4.2.3|" \
    -e "s|$SRCDIR/pkgs/itcl4.2.3/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/itcl4.2.3|/usr/include|"            \
    -i pkgs/itcl4.2.3/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 ../tcl8.6.13-html.tar.gz --strip-components=1
mkdir -v -p /usr/share/doc/tcl-8.6.13
cp -v -r  ./html/* /usr/share/doc/tcl-8.6.13
```

8.15.2. 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

8.16. Expect-5.45.4

Expect pakken inneholder verktøy for å automatisere, via skriptede dialoger, interaktive applikasjoner som f.eks **telnet**, **ftp**, **passwd**, **fsck**, **rlogin**, og **tip**. Expect er også nyttig for å teste disse samme applikasjoner i tillegg til å lette alle slags oppgaver som er uoverkommelige vanskelig med noe annet. DejaGnu rammeverket er skrevet i Expect.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 3.9 MB

8.16.1. Installasjon av Expect

Forbered Expect for kompilering:

```
./configure --prefix=/usr           \
            --with-tcl=/usr/lib     \
            --enable-shared         \
            --mandir=/usr/share/man \
            --with-tclinclude=/usr/include
```

Betydningen av konfigureringsalternativene:

`--with-tcl=/usr/lib`
Denne parameteren er nødvendig for å fortelle **configure** hvor **tclConfig.sh** skriptet er plassert.

`--with-tclinclude=/usr/include`
Dette forteller Expect eksplisitt hvor du finner Tcls interne deklarasjoner.

Bygg pakken:

```
make
```

For å teste resultatene, utsted:

```
make test
```

Installer pakken:

```
make install
ln -svf expect5.45.4/libexpect5.45.4.so /usr/lib
```

8.16.2. Innhold i Expect

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

Korte beskrivelser

expect Kommuniserer med andre interaktive programmer iht. til et skript

`libexpect-5.45.4.so` Inneholder funksjoner som gjør at Expect kan brukes som en Tcl utvidelse eller brukes direkte fra C eller C++ (uten Tcl)

8.17. DejaGNU-1.6.3

DejaGnu pakken inneholder et rammeverk for å kjøre test pakker på GNU verktøy. Det er skrevet i **expect**, som selv bruker Tcl verktøykommandospråk (Tool Command Language).

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 6.9 MB

8.17.1. Installasjon av DejaGNU

Upstream anbefaler å bygge DejaGNU i en dedikert byggemappe :

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

Forbered DejaGNU for kompilering:

```
../configure --prefix=/usr
makeinfo --html --no-split -o doc/dejagnu.html ../doc/dejagnu.texi
makeinfo --plaintext       -o doc/dejagnu.txt  ../doc/dejagnu.texi
```

Bygg og installer pakken:

```
make install
install -v -dm755  /usr/share/doc/dejagnu-1.6.3
install -v -m644   doc/dejagnu.{html,txt} /usr/share/doc/dejagnu-1.6.3
```

For å teste resultatene, utsted:

```
make check
```

8.17.2. Innhold i DejaGNU

Installert program: dejagnu og runtest

Korte beskrivelser

dejagnu DejaGNU hjelpekommandostarter

runtest Et innpakningsskript som lokaliserer riktig **expect** skall og kjører deretter DejaGNU

8.18. Binutils-2.40

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

Omtrentlig byggetid: 2.2 SBU
Nødvendig diskplass: 2.6 GB

8.18.1. Installasjon av Binutils

Kontroller at PTYene fungerer som de skal inne i chroot miljøet ved å utføre en enkel test:

```
expect -c "spawn ls"
```

Denne kommandoen burde gi følgende utdata:

```
spawn ls
```

Hvis utdataene i stedet inkluderer meldingen nedenfor, er miljøet ikke satt opp for riktig PTY drift. Dette problemet må løses før testpakkene for Binutils og GCC kjøres:

```
The system has no more ptys.
Ask your system administrator to create more.
```

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-gold       \
             --enable-ld=default \
             --enable-plugins    \
             --enable-shared     \
             --disable-werror    \
             --enable-64-bit-bfd \
             --with-system-zlib
```

Betydningen av konfigureringsparametrene:

`--enable-gold`
Bygg gold linkeren og installer den som ld.gold (ved siden av standard linker).

`--enable-ld=default`
Bygg den originale bfd linkeren og installer den som både ld (som er standard linker) og ld.bfd.

`--enable-plugins`
Aktiverer støtte for programtillegg for linkeren.

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

`--with-system-zlib`
Bruk det installerte zlib biblioteket i stedet for å bygge den inkluderte versjonen.

Kompiler pakken:

```
make tooldir=/usr
```

Betydningen av make parameteren:

`tooldir=/usr`

Vanligvis er 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).

Important

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')
```

Tolv tester mislykkes i gold testpakken når `--enable-default-pie` og `--enable-default-ssp` alternativene sendes til GCC.

Installer pakken:

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

Fjern ubrukelige statiske biblioteker og tom manside:

```
rm -fv /usr/lib/lib{bfd,ctf,ctf-nobfd,sframe,opcodes}.a
rm -fv /usr/share/man/man1/{gprofng,gp-*}.1
```

8.18.2. Innhold i Binutils

Installerte programmer:	addr2line, ar, as, c++filt, dwp, elfedit, gprof, gprofng, ld, ld.bfd, ld.gold, nm, objcopy, objdump, ranlib, readelf, size, strings, og strip
Installerte biblioteker:	libbfd.so, libctf.so, libctf-nobfd.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 kjørbar fil, bruker den feilsøkingsinformasjonen 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 linkeren 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 kallgrafen
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.gold	En nedskalert versjon av ld som bare støtter objektfil formatet elf
ld.bfd	Hardlenke til **ld**
nm	Lister symbolene som forekommer i en gitt objektfil
objcopy	Oversetter én type objektfil til en annen
objdump	Viser informasjon om den gitte objektfilen, med alternativer kontrollerer den hvilken informasjonen som skal vises; informasjonen som vises er nyttig for programmerere som jobber med kompileringens verktøy
ranlib	Genererer en indeks over innholdet i et arkiv og lagrer det i arkivet; indeksen viser alle symbolene som er definert av arkivmedlemmer som er flyttbare objektfiler
readelf	Viser informasjon om binærfiler av ELF typen
size	Viser seksjonsstørrelsene og totalstørrelsen for de gitte objektfilene
strings	Utdata, for hver gitt fil, sekvensene av utskrivbare tegn som er av minst den angitte lengden (som standard til fire); for objektfiler skriver den som standard bare strengene fra initialiserings- og lastingsseksjonene mens for andre typer filer, skanner den hele filen
strip	Fjerner symboler fra objektfiler
`libbfd`	Biblioteket med binære filbeskrivelser
`libctf`	Compat ANSI-C Type Format støttebibliotek for feilsøking
`libctf-nobfd`	En libctf variant som ikke bruker libbfd funksjonalitet
`libopcodes`	Et bibliotek for å håndtere opkoder—“leselig tekst” versjoner av instruksjoner for prosessoren; den brukes til å bygge verktøy som **objdump**
`libsframe`	Et bibliotek for å støtte tilbakesporing på nettet ved hjelp av en enkel avvikling

8.19. GMP-6.2.1

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

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 52 MB

8.19.1. Installasjon av GMP

Note

Hvis du bygger for 32-bit x86, men du har en CPU som er i stand til å kjøre 64-bits kode *og* du har spesifisert `CFLAGS` i miljøet vil konfigureringsskriptet forsøk å konfigurere for 64-biter og mislykkes. Unngå dette ved å påkalle configure kommandoen nedenfor med

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

Note

Standardinnstillingene til GMP produserer biblioteker optimalisert for vertsprosessoren. Hvis det er ønskelig med biblioteker egnet for prosessorer mindre kapable enn vertens CPU, kan generiske biblioteker bli opprettet ved å kjøre følgende:

```
cp -v configfsf.guess config.guess
cp -v configfsf.sub   config.sub
```

Forbered GMP for kompilering:

```
./configure --prefix=/usr    \
            --enable-cxx     \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.2.1
```

Betydningen av de nye konfigureringsalternativene:

`--enable-cxx`
Denne parameteren aktiverer C++ støtte

`--docdir=/usr/share/doc/gmp-6.2.1`
Denne variabelen spesifiserer riktig sted for dokumentasjon.

Kompiler pakken og generer HTML dokumentasjonen:

```
make
make html
```

Important

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

Test resultatene:

```
make check 2>&1 | tee gmp-check-log
```

Caution

Koden i gmp er svært optimalisert for prosessoren hvor den er bygget. Noen ganger vil koden som oppdager prosessoren feilidentifisere systemets evner og det vil være feil i testene eller andre applikasjoner som bruker gmp bibliotekene med meldingen "Illegal instruction". I dette tilfellet bør gmp rekonfigureres med alternativet --build=x86_64-pc-linux-gnu og gjenoppbygges.

Sørg for at alle 197 testene i testpakken besto. Sjekk resultatene ved å gi følgende kommando:

```
awk '/# PASS:/{total+=$3} ; END{print total}' gmp-check-log
```

Installer pakken og dens dokumentasjon:

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

8.19.2. Innhold i GMP

Installerte biblioteker: libgmp.so og libgmpxx.so
Installert mappe: /usr/share/doc/gmp-6.2.1

Korte beskrivelser

`libgmp` Inneholder matematiske presisjonsfunksjoner

`libgmpxx` Inneholder C++ matematiske presisjonsfunksjoner

8.20. MPFR-4.2.0

MPFR pakken inneholder matematiske funksjoner med flere presisjoner.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 43 MB

8.20.1. Installasjon av MPFR

Rett opp en testsak basert på en feil med gamle Glibc utgivelser:

```
sed -e 's/+01,234,567/+1,234,567 /' \
    -e 's/13.10Pd/13Pd/'            \
    -i tests/tsprintf.c
```

Forbered MPFR for kompilering:

```
./configure --prefix=/usr        \
            --disable-static     \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.2.0
```

Kompiler pakken og generer HTML dokumentasjonen:

```
make
make html
```

Important

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

Test resultatene og sørg for at alle 197 testene består:

```
make check
```

Installer pakken og dens dokumentasjon:

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

8.20.2. Innhold i MPFR

Installerte biblioteker: libmpfr.so
Installert mappe: /usr/share/doc/mpfr-4.2.0

Korte beskrivelser

`libmpfr` Inneholder matematiske funksjoner med flere presisjoner

8.21. 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

8.21.1. 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
```

8.21.2. Innhold i MPC

Installerte biblioteker: libmpc.so
Installert mappe: /usr/share/doc/mpc-1.3.1

Korte beskrivelser

`libmpc` Inneholder komplekse matematiske funksjoner

8.22. Attr-2.5.1

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

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 4.1 MB

8.22.1. Installasjon av Attr

Forbered Attr for kompilering:

```
./configure --prefix=/usr     \
            --disable-static  \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.5.1
```

Kompiler pakken:

```
make
```

Testene må kjøres på et filsystem som støtter utvidete attributter som filsystemene ext2, ext3 eller ext4. For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.22.2. Innhold i Attr

Installerte programmer: attr, getfattr, og setfattr
Installert bibliotek: libattr.so
Installerte mapper: /usr/include/attr og /usr/share/doc/attr-2.5.1

Korte beskrivelser

attr Utvider attributter på filsystemobjekter

getfattr Henter de utvidede attributtene til filsystemobjekter

setfattr Angir de utvidede attributtene til filsystemobjekter

`libattr` Inneholder bibliotekfunksjonene for å manipulere utvidede attributter

8.23. Acl-2.3.1

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

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 6.1 MB

8.23.1. Installasjon av Acl

Forbered Acl for kompilering:

```
./configure --prefix=/usr         \
            --disable-static      \
            --docdir=/usr/share/doc/acl-2.3.1
```

Kompiler pakken:

```
make
```

Acl testene må kjøres på et filsystem som støtter tilgangskontroller, men ikke før Coreutils pakken er bygget, ved å bruke Acl biblioteker. Hvis ønskelig, gå tilbake til denne pakken og kjør **make check** etter at Coreutils pakken er bygget.

Installer pakken:

```
make install
```

8.23.2. Innhold i Acl

Installerte programmer: chacl, getfacl, og setfacl
Installert bibliotek: libacl.so
Installerte mapper: /usr/include/acl og /usr/share/doc/acl-2.3.1

Korte beskrivelser

chacl Endrer tilgangskontrolllisten til en fil eller mappe

getfacl Henter filtilgangskontrolllister

setfacl Angir filtilgangskontrolllister

`libacl` Inneholder bibliotekfunksjonene for å manipulere tilgangskontrolllister

8.24. Libcap-2.67

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: mindre enn 0.1 SBU
Nødvendig diskplass: 2.9 MB

8.24.1. Installasjon av Libcap

Hindre at statiske biblioteker blir installert:

```
sed -i '/install -m.*STA/d' libcap/Makefile
```

Kompiler pakken:

```
make prefix=/usr lib=lib
```

Betydningen av make alternativet:

`lib=lib`
Denne parameteren setter 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
```

8.24.2. Innhold i Libcap

Installerte programmer: capsh, getcap, getpcaps, og setcap
Installert bibliotek: libcap.so og libpsx.so

Korte beskrivelser

capsh En skallinnpakning for å utforske og begrense funksjonsstøtte
getcap Undersøker filfunksjoner
getpcaps Viser egenskapene til de forespurte prosessen(e)
setcap Angir filfunksjoner
`libcap` Inneholder bibliotekfunksjonene for å manipulere POSIX 1003.1e funksjoner
`libpsx` Inneholder funksjoner for å støtte POSIX semantikk for syscalls knyttet til pthread biblioteket

8.25. Shadow-4.13

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

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 46 MB

8.25.1. Installasjon av Shadow

Note

Hvis du ønsker å håndheve bruken av sterke passord, se *https://www.linuxfromscratch.org/blfs/view/11.3/postlfs/cracklib.html* for installasjon av CrackLib før du bygger Shadow. Legg så til `--with-libcrack` til **configure** kommandoen under.

Deaktiver installasjonen av **groups** programmet og mansidene, ettersom Coreutils gir en bedre versjon. Også, forhindre installasjon av mansider som allerede var installert i Section 8.3, “Man-pages-6.03”:

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

I stedet for å bruke standard *crypt* metoden, bruk den sikrere *SHA-512* metoden for passordkryptering, som også tillater passord lengre enn 8 tegn. I tillegg, angi antall avrundinger til 500 000 i stedet for standard 5000, som er altfor lavt til å forhindre brute force passordangrep. Det er også nødvendig å endre det foreldede `/var/spool/mail` plasseringen for brukerpostbokser som Shadow bruker som standard til `/var/mail` stedet som brukes for øyeblikket. Og, fjerne `/bin` og `/sbin` fra `PATH`, siden de ganske enkelt er symbolske lenker til motparten i `/usr`.

Note

Hvis du ønsker å inkludere `/bin` og/eller `/sbin` i `PATH` av en eller annen grunn, endre `PATH` i `.bashrc` etter at LFS er bygget.

```
sed -e 's:#ENCRYPT_METHOD DES:ENCRYPT_METHOD SHA512:' \
    -e 's@#\(SHA_CRYPT_..._ROUNDS 5000\)@\100@'       \
    -e 's:/var/spool/mail:/var/mail:'                 \
    -e '/PATH=/{s@/sbin:@@;s@/bin:@@}'                \
    -i etc/login.defs
```

Note

Hvis du valgte å bygge Shadow med støtte for Cracklib, kjør følgende:

```
sed -i 's:DICTPATH.*:DICTPATH\t/lib/cracklib/pw_dict:' etc/login.defs
```

Forbered Shadow for kompilering:

```
touch /usr/bin/passwd
./configure --sysconfdir=/etc \
            --disable-static  \
            --with-group-name-max-length=32
```

Betydningen av konfigureringsalternativet:

touch /usr/bin/passwd
Filen `/usr/bin/passwd` må eksistere fordi plasseringen er hardkodet i noen programmer; hvis den ikke eksisterer allerede, vil installasjonsskriptet lage den på feil sted.

`--with-group-name-max-length=32`
Det lengste tillatte brukernavnet er 32 tegn. Gjør det maksimale lengden på et gruppenavn til det samme.

Kompiler pakken:

```
make
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

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

8.25.2. Konfigurerer Shadow

Denne pakken inneholder verktøy for å legge til, endre og slette brukere og grupper; angi og endre passordene deres; og utføre annen administrativ oppgaver. For en fullstendig forklaring av hva *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 Section 7.6, “Opprette essensielle filer og symbolkoblinger”.

`CREATE_MAIL_SPOOL=yes`

Denne parameteren gjør at **useradd** lager en postboksfil for den nyopprettede brukeren. **useradd** vil gjøre gruppe eierskap av denne filen til `mail` gruppe med 0660 tillatelser. Hvis du ikke vil opprette disse filene, gi følgende kommando:

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

8.25.3. Sette root passordet

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

```
passwd root
```

8.25.4. Innhold i Shadow

Installerte programmer:	chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, getsubids, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgidmap, newgrp, newuidmap, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (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
chgpasswd	Brukes til å oppdatere gruppepassord i skriptmodus
chpasswd	Brukes til å oppdatere brukerpassord i skriptmodus
chsh	Brukes til å endre en brukers standard påloggingsskall
expiry	Sjekker og håndhever gjeldende retningslinjer for passordutløp
faillog	Brukes til å undersøke loggen over påloggingsfeil, for å sette et maksimum antall feil før en konto blokkeres, eller for å tilbakestille antall feil
getsubids	Brukes til å liste de underordnede id-områdene for en bruker
gpasswd	Brukes til å legge til og slette medlemmer og administratorer til grupper
groupadd	Oppretter en gruppe med det gitte navnet
groupdel	Sletter gruppen med oppgitt navn
groupmems	Lar en bruker administrere sin egen gruppemedlemsliste uten krav om superbrukerprivilegier.
groupmod	Brukes til å endre den gitte gruppens navn eller GID
grpck	Verifiserer integriteten til gruppefilene `/etc/group` og `/etc/gshadow`
grpconv	Oppretter eller oppdaterer skyggegruppefilen fra den normale gruppefilen
grpunconv	Oppdaterer `/etc/group` fra `/etc/gshadow` og sletter deretter sistnevnte
lastlog	Rapporterer siste pålogging for alle brukere eller av en gitt bruker

login Brukes av systemet for å la brukere logge på

logoutd Er en nisse som brukes til å håndheve restriksjoner på påloggingstid og portene

newgidmap Brukes til å angi gid-tilordning av et brukernavnområde

newgrp Brukes til å endre gjeldende GID under en påloggingsøkt

newuidmap Brukes til å angi uid-tilordning av et brukernavnområde

newusers Brukes til å opprette eller oppdatere en hel serie med brukerkontoer

nologin Viser en melding 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 `/etc/passwd` og `/etc/shadow`

pwconv Oppretter eller oppdaterer skyggepassordfilen fra den normale passordfilen

pwunconv Oppdaterer `/etc/passwd` fra `/etc/shadow` og sletter deretter sistnevnte

sg Utfører en gitt kommando mens brukerens GID er satt til den gitte gruppen

su Kjører et skall med erstatningsbruker- og gruppe-IDer

useradd Oppretter en ny bruker med det gitte navnet, eller oppdaterer standard informasjon om ny bruker

userdel Sletter den gitte brukerkontoen

usermod Brukes til å endre den gitte brukerens påloggingsnavn, bruker identifikasjon (UID), skall, startgruppe, hjemmekatalog, etc.

vigr Redigerer `/etc/group` eller `/etc/gshadow` filene

vipw Redigerer `/etc/passwd` eller `/etc/shadow` filene

`libsubid` bibliotek for å prosessere underordnede id-områder for brukere

8.26. GCC-12.2.0

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

Omtrentlig byggetid: 43 SBU (med tester)
Nødvendig diskplass: 5.1 GB

8.26.1. 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     \
             --disable-multilib       \
             --disable-bootstrap      \
             --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 konfigureringsskriptet bruker ld installert av binutils bygget tidligere i dette kapittelet, i stedet for den kryssbygde versjonen som ellers ville blitt brukt.

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

Note

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 filen. 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. Stabelkorrupsjon 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
```

Important

I denne delen vurderes testpakken for GCC å være viktig, men det tar lang tid. Førstegangsbyggere oppfordres til ikke å hoppe over dette. Tiden for å kjøre testene kan bli redusert betydelig ved å legge til -jx til **make -k check** kommandoen nedenfor hvor x er antall kjerner på systemet ditt.

Et sett med tester i GCC testpakken er kjent for å bruke opp standard stabel (stack), så øk stabelstørrelsen før du kjører testene:

```
ulimit -s 32768
```

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

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

For å 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/11.3/* og *https://gcc.gnu.org/ml/gcc-testresults/*.

I gcc er elleve tester i i386-testpakken kjent for å FEILE. Det er fordi testfilene ikke tar hensyn til *`--enable-default-pie`* alternativet.

Fire tester relatert til PR100400 kan rapporteres som både XPASS og FAIL når du tester g++-kompilatoren; testfilen er ikke godt skrevet.

Noen få uventede feil kan ikke alltid unngås. GCC utviklerne er vanligvis klar over disse problemene, men har ikke løst dem ennå. Med mindre testresultatene er svært forskjellige fra de på URLen ovenfor, er det trygt å fortsette.

Installer pakken:

```
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)/12.2.0/include{,-fixed}
```

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

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

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

```
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/12.2.0/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[1in].*succeeded' dummy.log
```

Utdata fra den siste kommandoen bør være:

```
/usr/lib/gcc/x86_64-pc-linux-gnu/12.2.0/../../../../lib/Scrt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/12.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/12.2.0/../../../../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/12.2.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/12.2.0/include-fixed
 /usr/include
```

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

Deretter bekrefter du at den nye linkeren brukes med de riktige søkebanene:

```
grep 'SEARCH.*/usr/lib' dummy.log |sed 's|; |\n|g'
```

Referanser til stier som har komponenter med '-linux-gnu' bør ignoreres, men ellers 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
```

8.26.2. Innhold i GCC

Installerte programmer:	c++, 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}, libitm.{a,so}, liblsan.{a,so}, liblto_plugin.so, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so}, libstdc++fs.a, libsupc++.a, libtsan.{a,so}, og libubsan.{a,so}
Installerte mapper:	/usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc, og /usr/share/gcc-12.2.0

Korte beskrivelser

c++	C++ kompilatoren
cc	C kompilatoren
cpp	C forprosessoren; den brukes av kompilatoren for å utvide #include, #define og lignende utsagn i kildefilene
g++	C++ kompilatoren
gcc	C kompilatoren
gcc-ar	En innpakning rundt **ar** som legger til et programtillegg til kommandolinjen. Dette programmet brukes kun 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 dekningstestverktøy; den brukes til å analysere programmer å bestemme hvor optimaliseringer vil ha størst effekt

gcov-dump Frakoblet (offline) gcda og gcno profildumpverktøy

gcov-tool Frakoblet (offline) gcda profilbehandlingsverktøy

lto-dump Verktøy for dumping av objektfiler produsert av GCC med LTO aktivert

`libasan` Kjøretidsbiblioteket for adresserensing

`libatomic` GCC atomic innebygde kjøretidsbibliotek

`libcc1` C forbehandlingsbiblioteket

`libgcc` Inneholder kjøretidsstøtte for **gcc**

`libgcov` Dette biblioteket er koblet inn i et program når GCC blir instruert om å aktivere profilering

`libgomp` GNU implementering av OpenMP API for multiplattform parallellprogrammering med delt minne i C/C++ og Fortran

`libitm` GNU transaksjonsminnebiblioteket

`liblsan` Leak Sanitizer kjøretidsbibliotek

`liblto_plugin` GCC sitt LTO programtillegg som lar binutils behandle objektfiler produsert av GCC med LTO aktivert

`libquadmath` GCC Quad Precision Math Library API

`libssp` Inneholder rutiner som støtter GCCs stabelknusende beskyttelsesfunksjonalitet. Normalt er det ubrukt fordi glibc også gir disse rutinene

`libstdc++` Standard C++ biblioteket

`libstdc++fs` ISO/IEC TS 18822:2015 filsystembibliotek

`libsupc++` Gir støttende rutiner for C++ programmeringsspråk

`libtsan` Thread Sanitizer kjøretidsbibliotek

`libubsan` Undefined Behavior Sanitizer kjøretidsbibliotek

8.27. Pkg-config-0.29.2

Pakken pkg-config inneholder et verktøy for å sende inkluderingsbanen og/eller bibliotekstier for å bygge verktøy under konfigurerings- og makefasene av pakkeinstallasjoner.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 29 MB

8.27.1. Installasjon av Pkg-config

Forbered Pkg-config for kompilering:

```
./configure --prefix=/usr              \
            --with-internal-glib       \
            --disable-host-tool        \
            --docdir=/usr/share/doc/pkg-config-0.29.2
```

Betydningen av de nye konfigureringsalternativene:

`--with-internal-glib`
Dette vil tillate pkg-config å bruke sin interne versjon av Glib fordi en ekstern versjon ikke er tilgjengelig i LFS.

`--disable-host-tool`
Dette alternativet deaktiverer opprettelsen av en uønsket hard lenke til pkg-config programmet.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.27.2. Innhold i Pkg-config

Installert program: pkg-config
Installert mappe: /usr/share/doc/pkg-config-0.29.2

Korte beskrivelser

pkg-config Returnerer metainformasjon for det angitte biblioteket eller pakken

8.28. Ncurses-6.4

Ncurses pakken inneholder biblioteker for terminaluavhengig håndtering av karakterskjermer.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 45 MB

8.28.1. Installasjon av Ncurses

Forbered Ncurses for kompilering:

```
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            --with-shared           \
            --without-debug         \
            --without-normal        \
            --with-cxx-shared       \
            --enable-pc-files       \
            --enable-widec          \
            --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.

`--enable-widec`
Denne bryteren forårsaker at biblioteker med brede tegn (f.eks., `libncursesw.so.6.4`) blir bygget i stedet for vanlige (f.eks., `libncurses.so.6.4`). Disse bibliotekene med brede tegn er brukbare i både multibyte og tradisjonelle 8-biters lokaliteter, mens vanlige biblioteker fungerer som de skal bare i 8-biters lokaliteter. Biblioteker med brede tegn og normale biblioteker er kildekompatibel, men ikke binærkompatibel.

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.6.4` . Det kan krasje skallprosessen som bruker kode og data fra bibliotekfilen. Installer pakken med `DESTDIR`, og bytt ut bibliotekfilen riktig med **install** kommandoen:

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/libncursesw.so.6.4 /usr/lib
rm -v  dest/usr/lib/libncursesw.so.6.4
cp -av dest/* /
```

Mange applikasjoner forventer fortsatt at lenkeren skal kunne finne Ncurses biblioteker med ikke-brede karakterer. Lur slike applikasjoner til å koble til biblioteker med brede tegn ved hjelp av symbolkoblinger og lenkerskript:

```
for lib in ncurses form panel menu ; do
    rm -vf                    /usr/lib/lib${lib}.so
    echo "INPUT(-l${lib}w)" > /usr/lib/lib${lib}.so
    ln -sfv ${lib}w.pc        /usr/lib/pkgconfig/${lib}.pc
done
```

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

```
rm -vf                     /usr/lib/libcursesw.so
echo "INPUT(-lncursesw)" > /usr/lib/libcursesw.so
ln -sfv libncurses.so      /usr/lib/libcurses.so
```

Hvis ønskelig, installer Ncurses dokumentasjonen:

```
mkdir -pv      /usr/share/doc/ncurses-6.4
cp -v -R doc/* /usr/share/doc/ncurses-6.4
```

Note

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
```

8.28.2. Innhold i Ncurses

Installerte programmer: captoinfo (lenker til tic), clear, infocmp, infotocap (lenker til tic), ncursesw6-config, reset (lenker til tset), tabs, tic, toe, tput, og tset

Installerte biblioteker: libcursesw.so (symlenke og lenkerskript til libncursesw.so), libformw.so, libmenuw.so, libncursesw.so, libncurses++w.so, libpanelw.so, og deres ikke-brede karaktermotstykker uten "w" i biblioteknavnene.

Installerte mapper: /usr/share/tabset, /usr/share/terminfo, og /usr/share/doc/ncurses-6.4

Korte beskrivelser

captoinfo Konverterer en termcap beskrivelse til en terminfo beskrivelse

clear Tømmer skjermen hvis mulig

infocmp Sammenligner eller skriver ut terminfo beskrivelser

infotocap Konverterer en terminfo beskrivelse til en termcap beskrivelse

ncursesw6-config Gir konfigurasjonsinformasjon for ncurses

reset	Reinitialiserer en terminal til standardverdiene
tabs	Fjerner og setter tabulatorstopp på en terminal
tic	Terminfo entry-description kompilatoren som oversetter en terminfo fil fra kildeformat til det binære formatet som trengs for ncurses biblioteksrutiner [En terminfo fil inneholder informasjon om egenskapene til en bestemt terminal.]
toe	Viser alle tilgjengelige terminaltyper, med primærnavn og beskrivelse for hver
tput	Gjør verdiene til terminalavhengige funksjoner tilgjengelig for skallet; den kan også brukes til å tilbakestille eller initialisere en terminal eller rapporter det lange navnet
tset	Kan brukes til å initialisere terminaler
`libcursesw`	En lenke til `libncursesw`
`libncursesw`	Inneholder funksjoner for å vise tekst på mange komplekse måter på en terminalskjerm; et godt eksempel på bruken av disse funksjonene er menyen som vises under kjernens **make menuconfig**
`libncurses++w`	Inneholder C++ binding for andre biblioteker i denne pakken
`libformw`	Inneholder funksjoner for å implementere skjemaer
`libmenuw`	Inneholder funksjoner for å implementere menyer
`libpanelw`	Inneholder funksjoner for å implementere paneler

8.29. Sed-4.9

Sed pakken inneholder en dataflyt (stream) redigerer.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 31 MB

8.29.1. Installation of Sed

Forbered Sed for kompilering:

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

Kompiler pakken og generer HTML dokumentasjonen:

```
make
make html
```

For å teste resultatene, utsted:

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

Installer pakken og dens dokumentasjon:

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

8.29.2. 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

8.30. Psmisc-23.6

Psmisc pakken inneholder programmer for å vise informasjon om kjørende prosesser.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 6.5 MB

8.30.1. Installasjon av Psmisc

Forbered Psmisc for kompilering:

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

Kompiler pakken:

```
make
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make install
```

8.30.2. Innhold i Psmisc

Installerte programmer: fuser, killall, peekfd, prtstat, pslog, pstree, og pstree.x11 (lenker til pstree)

Korte beskrivelser

fuser Rapporterer prosessIDene (PIDene) til prosesser som bruker de gitte filer eller filsystemer

killall Dreper prosesser ved navn; den sender et signal til alle prosesser som kjører noen av de gitte kommandoene

peekfd Se på filbeskrivelser for en prosess som kjører, gitt dens PID

prtstat Skriver ut informasjon om en prosess

pslog Rapporterer gjeldende loggbane for en prosess

pstree Viser kjørende prosesser som et tre

pstree.x11 Samme som **pstree**, bortsett fra at den venter på bekreftelse før den avslutter

8.31. Gettext-0.21.1

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: 241 MB

8.31.1. Installasjon av Gettext

Forbered Gettext for kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.21.1
```

Kompiler pakken:

```
make
```

For å teste resultatene (dette tar lang tid, rundt 3 SBU), utsted:

```
make check
```

Installer pakken:

```
make install
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

8.31.2. Innhold i Gettext

Installerte programmer: autopoint, envsubst, gettext, gettext.sh, gettextize, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin, og xgettext
Installerte biblioteker: libasprintf.so, libgettextlib.so, libgettextpo.so, libgettextsrc.so, libtextstyle.so, og preloadable_libintl.so
Installerte mapper: /usr/lib/gettext, /usr/share/doc/gettext-0.21.1, /usr/share/gettext, og /usr/share/gettext-0.19.8

Korte beskrivelser

autopoint Kopierer standard Gettext infrastrukturfiler til en kildepakke

envsubst Erstatter miljøvariabler i skallformatstrenger

gettext Oversetter en melding på det opprinnelige språket til brukerens språk ved å slå opp oversettelsen i en meldingskatalog

gettext.sh Fungerer først og fremst som et skallfunksjonsbibliotek for gettext

gettextize Kopierer alle standard Gettext filer til den gitte mappens toppnivå til en pakke for å begynne å internasjonalisere den

msgattrib Filtrerer meldingene i en oversettelsesmappe i henhold til deres attributter og manipulerer attributtene

msgcat Sammenslår og slår sammen de gitte `.po` filene

msgcmp	Sammenligner to `.po` filer for å sjekke at begge inneholder samme sett med msgid strenger
msgcomm	Finner meldingene som er felles for de gitte `.po` filene
msgconv	Konverterer en oversettelseskatalog til en annet tegnkoding
msgen	Oppretter en engelsk oversettelseskatalog
msgexec	Bruker en kommando på alle oversettelser av en oversettelsesmappe
msgfilter	Bruker et filter på alle oversettelser av en oversettelsesmappe
msgfmt	Genererer en binær 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 `.po` fil, initialisere metainformasjonen med verdier fra brukerens miljø
msgmerge	Kombinerer to rå oversettelser til én enkelt fil
msgunfmt	Dekompilerer en binær meldingskatalog til rå oversettelsestekst
msguniq	Forener dupliserte oversettelser i en oversettelsesmappe
ngettext	Viser oversettelser på morsmål av en tekstmelding hvis grammatisk form avhenger av et tall
recode-sr-latin	Omkoder serbisk tekst fra kyrillisk til latinsk skrift
xgettext	Trekker ut de oversettbare meldingslinjene fra den gitte kildefilen for å lage den første oversettelsesmalen
`libasprintf`	Definerer *autosprintf* klassen, som gir C formaterte utdatarutiner som kan brukes i C++ programmer, for bruk med <*string*> strenger og <*iostream*> dataflyt
`libgettextlib`	Inneholder vanlige rutiner som brukes av ulike Gettext programmer; disse er ikke beregnet for generelt bruk
`libgettextpo`	Brukes til å skrive spesialiserte programmer som behandler `.po` filer; dette biblioteket brukes når standardapplikasjonene som ble levert med Gettext (som f.eks **msgcomm**, **msgcmp**, **msgattrib**, og **msgen**) ikke er tilstrekkelig
`libgettextsrc`	Gir vanlige rutiner som brukes av ulike Gettext programmer; disse er ikke beregnet for generelt bruk
`libtextstyle`	Tekststilbibliotek
`preloadable_libintl`	Et bibliotek, ment å brukes av LD_PRELOAD som assisterer `libintl` i å logge uoversatte meldinger

8.32. Bison-3.8.2

Bison pakken inneholder en parsergenerator.

Omtrentlig byggetid: 2.3 SBU
Nødvendig diskplass: 62 MB

8.32.1. Installasjon av Bison

Forbered Bison for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene (about 5.5 SBU), utsted:

```
make check
```

Installer pakken:

```
make install
```

8.32.2. Innholdet i Bison

Installerte programmer: bison og yacc
Installerte biblioteker: liby.a
Installert katalog: /usr/share/bison

Korte beskrivelser

bison Genererer, fra en rekke regler, et program for å analysere struktur av tekstfiler; Bison er en erstatning for Yacc (Yet Another Compiler Compiler)

yacc En innpakning for **bison**,ment for programmer som fortsatt kaller **yacc** i stedet for **bison**; den kaller **bison** med *-y* alternativet

`liby` Yacc biblioteket som inneholder implementeringer av Yacc kompatibel `yyerror` og `main` funksjoner; dette biblioteket er normalt lite nyttig, men POSIX krever det

8.33. Grep-3.8

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

Omtrentlig byggetid: 0.4 SBU
Nødvendig diskplass: 37 MB

8.33.1. 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
```

8.33.2. 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

8.34. Bash-5.2.15

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

Omtrentlig byggetid: 1.2 SBU
Nødvendig diskplass: 52 MB

8.34.1. Installasjon av Bash

Forbered Bash for kompilering:

```
./configure --prefix=/usr             \
            --without-bash-malloc     \
            --with-installed-readline \
            --docdir=/usr/share/doc/bash-5.2.15
```

Betydningen av det nye konfigureringsalternativet:

`--with-installed-readline`
Dette alternativet forteller Bash å bruke `readline` biblioteket som allerede er installert på systemet i stedet for å bruke sin egen readline versjon.

Kompiler pakken:

```
make
```

Hopp ned til “Installer pakken” hvis du ikke kjører testpakken.

For å forberede testene, sørg for at brukeren `tester` kan skrive til kildetreet:

```
chown -Rv tester .
```

Testpakken til pakken er designet for å kjøres som en ikke-`root` bruker som eier terminalen koblet til standardinngang. For å tilfredsstille 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** får 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
```

8.34.2. 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-5.2.15

Korte beskrivelser

bash En mye brukt kommandotolk; den utfører mange typer av utvidelser og erstatninger på en gitt kommandolinje før kjøringen gjøres , og dette gjør dermed denne tolken til et kraftig verktøy

bashbug Et skallskript for å hjelpe brukeren med å skrive og sende standard formaterte feilrapporter vedrørende **bash**

sh En symbolsk lenke til **bash** programmet; når det påkalles som **sh**, prøver **bash** å etterligne oppstartadferd av historiske versjoner av **sh** så nært som mulig, samtidig som den også samsvarer med POSIX standarden

8.35. Libtool-2.4.7

Libtool pakken inneholder GNU generiske bibliotekstøtteskript. Det omslutter kompleksiteten ved å bruke delte biblioteker i en konsistent, overførbart grensesnitt.

Omtrentlig byggetid: 1.4 SBU
Nødvendig diskplass: 44 MB

8.35.1. Installasjon av Libtool

Forbered Libtool for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make -k check
```

Note

Testtiden for libtool kan reduseres betydelig på et system med flere kjerner. For å gjøre dette, legg til **TESTSUITEFLAGS=-j<N>** til linjen over. For eksempel kan bruk av -j4 redusere testtiden med over 60 prosent.

Fem tester er kjent for å mislykkes i LFS byggemiljøet pga en sirkulær avhengighet, men disse testene består hvis de sjekkes på nytt etter at automake er installert. I tillegg, med grep-3.8, vil to tester utløse en advarsel for ikke-POSIX regulære uttrykk og mislykkes.

Installer pakken:

```
make install
```

Fjern et ubrukelig statisk bibliotek:

```
rm -fv /usr/lib/libltdl.a
```

8.35.2. Innhold i Libtool

Installerte programmer: libtool og libtoolize
Installerte biblioteker: libltdl.so
Installerte mapper: /usr/include/libltdl og /usr/share/libtool

Korte beskrivelser

libtool Tilbyr generaliserte støttetjenester for bibliotekbygging

libtoolize Gir en standard måte å legge til **libtool** støtte til en pakke

`libltdl` Skjuler de ulike vanskelighetene med å åpne dynamisk lastede biblioteker

8.36. GDBM-1.23

GDBM pakken inneholder GNU Database Manager. Det er et bibliotek av databasefunksjoner som bruker utvidbar hashing og fungerer lignende som standard UNIX dbm. Biblioteket gir primitiver for lagring av nøkkel/data par, søker og henter dataene etter nøkkelen og sletter en nøkkel sammen med sine data.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 13 MB

8.36.1. Installasjon av GDBM

Forbered GDBM for kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --enable-libgdbm-compat
```

Betydningen av konfigureringsalternativet:

`--enable-libgdbm-compat`
Denne bryteren gjør det mulig å bygge libgdbm kompatibilitetsbiblioteket. Noen pakker utenfor LFS kan kreve eldre DBM rutiner det gir.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.36.2. Innhold i GDBM

Installerte programmer: gdbm_dump, gdbm_load, og gdbmtool
Installerte biblioteker: libgdbm.so og libgdbm_compat.so

Korte beskrivelser

gdbm_dump Dumper en GDBM database til en fil
gdbm_load Gjenoppretter en GDBM database fra en dumpfil
gdbmtool Tester og modifiserer en GDBM database
`libgdbm` Inneholder funksjoner for å manipulere en hashet database
`libgdbm_compat` Kompatibilitetsbibliotek som inneholder eldre DBM funksjoner

8.37. Gperf-3.1

Gperf genererer en perfekt hashfunksjon fra et nøkkelsett.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 6.1 MB

8.37.1. Installasjon av Gperf

Forbered Gperf for kompilering:

```
./configure --prefix=/usr --docdir=/usr/share/doc/gperf-3.1
```

Kompiler pakken:

```
make
```

Testene er kjent for å mislykkes hvis de kjører flere samtidige tester (-j alternativ større enn 1). Å teste resultatene, utsted:

```
make -j1 check
```

Installer pakken:

```
make install
```

8.37.2. Innhold i Gperf

Installert program: gperf
Installert mappe: /usr/share/doc/gperf-3.1

Korte beskrivelser

gperf Genererer en perfekt hash fra et nøkkelsett

8.38. Expat-2.5.0

Expat pakken inneholder et dataflytorientert C bibliotek for å analysere XML.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 12 MB

8.38.1. Installasjon av Expat

Forbered Expat for kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.5.0
```

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-2.5.0
```

8.38.2. Innhold i Expat

Installert program: xmlwf
Installerte biblioteker: libexpat.so
Installert mappe: /usr/share/doc/expat-2.5.0

Korte beskrivelser

xmlwf Er et ikke-validerende verktøy for å sjekke om XML dokumenter er godt utformet

`libexpat` Inneholder API funksjoner for å analysere XML

8.39. Inetutils-2.4

Inetutils pakken inneholder programmer for grunnleggende nettverksbygging.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 31 MB

8.39.1. Installasjon av Inetutils

Forbered Inetutils for kompilering:

```
./configure --prefix=/usr        \
            --bindir=/usr/bin    \
            --localstatedir=/var \
            --disable-logger     \
            --disable-whois      \
            --disable-rcp        \
            --disable-rexec      \
            --disable-rlogin     \
            --disable-rsh        \
            --disable-servers
```

Betydningen av konfigureringsalternativene:

`--disable-logger`
Dette alternativet forhindrer Inetutils fra å installere **logger** programmet, som brukes av skript til sende meldinger til Systemlogg nissen (System Log Daemon). Ikke installer det fordi Util-linux installerer en nyere versjon.

`--disable-whois`
Dette alternativet deaktiverer byggingen av Inetutils sin **whois** klient, som er utdatert. Instruksjoner for en bedre **whois** klienten er i BLFS boken.

`--disable-r*`
Disse parameterne deaktiverer bygging av foreldede programmer som ikke burde brukes på grunn av sikkerhetsproblemer. Funksjonene som tilbys av disse programmer kan leveres av openssh pakken i BLFS boken.

`--disable-servers`
Dette deaktiverer installasjonen av de forskjellige nettverksserverne inkludert som en del av Inetutils pakken. Disse serverne anses ikke som hensiktsmessig i et grunnleggende LFS system. Noen er usikre av natur og er ansett som trygt kun på pålitelige nettverk. Merk at bedre erstatninger er tilgjengelige for mange av disse serverne.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

Flytt et program til riktig plassering:

```
mv -v /usr/{,s}bin/ifconfig
```

8.39.2. Innhold i Inetutils

Installerte programmer: dnsdomainname, ftp, ifconfig, hostname, ping, ping6, talk, telnet, tftp, og traceroute

Korte beskrivelser

dnsdomainname	Vis systemets DNS domenenavn
ftp	Er protokollprogrammet for filoverføringer
hostname	Rapporterer eller angir navnet på verten
ifconfig	Administrerer nettverksgrensesnitter
ping	Sender ekkoforespørselspakker og rapporterer hvor 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

8.40. Less-608

Less pakken inneholder et tekstfilviser.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 4.3 MB

8.40.1. Installasjon av Less

Forbered Less for kompilering:

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

Betydningen av konfigureringsalternativene:

`--sysconfdir=/etc`
Dette alternativet forteller at programmene som er opprettet av pakken, skal se i `/etc` for konfigurasjonen filer.

Kompiler pakken:

```
make
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make install
```

8.40.2. Innhold av Less

Installerte programmer: less, lessecho, og lesskey

Korte beskrivelser

less En filviser eller søker; den viser innholdet i det gitte fil, lar brukeren rulle, finne strenger og hoppe til merker

lessecho Trengs for å utvide metakarakterer, som f.eks * og *?*, i filnavn på Unix systemer

lesskey Brukes til å spesifisere tastaturbindingene for **less**

8.41. Perl-5.36.0

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

Omtrentlig byggetid: 7.9 SBU
Nødvendig diskplass: 234 MB

8.41.1. 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                                         \
             -Dprefix=/usr                                \
             -Dvendorprefix=/usr                          \
             -Dprivlib=/usr/lib/perl5/5.36/core_perl      \
             -Darchlib=/usr/lib/perl5/5.36/core_perl      \
             -Dsitelib=/usr/lib/perl5/5.36/site_perl      \
             -Dsitearch=/usr/lib/perl5/5.36/site_perl     \
             -Dvendorlib=/usr/lib/perl5/5.36/vendor_perl  \
             -Dvendorarch=/usr/lib/perl5/5.36/vendor_perl \
             -Dman1dir=/usr/share/man/man1                \
             -Dman3dir=/usr/share/man/man3                \
             -Dpager="/usr/bin/less -isR"                 \
             -Duseshrplib                                 \
             -Dusethreads
```

Betydningen av konfigureringsalternativene:

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

`-Dpager="/usr/bin/less -isR"`
Dette sikrer at **`less`** brukes i stedet for **`more`**.

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`
Siden Groff ikke er installert ennå, vil ikke **Configure** opprette mansider for Perl. Disse parameterene overstyrer denne oppførselen.

`-Duseshrplib`
Bygger en delt libperl som trengs av noen perl moduler.

`-Dusethreads`
Bygg perl med støtte for tråder.

`-Dprivlib,-Darchlib,-Dsitelib,...`
Disse innstillingene definerer hvor Perl leter etter installerte moduler. LFS redaktørene valgte å legge dem i en katalogstruktur basert på Major.Minor-versjonen av Perl (5.36) hvilket tillater oppgradering av Perl til nyere Patch nivåer (Patchnivået er den siste punktseparerte delen i den fullstendige versjonenstrengen som 5.36.0) uten å installere alle modulene på nytt.

Kompiler pakken:

```
make
```

For å teste resultatene (ca. 11 SBU), utsted:

```
make test
```

Installer pakken og rydd opp:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

8.41.2. Innhold i Perl

Installerte programmer:	corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.36.0 (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
perl5.36.0	En hard lenke til **perl**
perlbug	Brukes til å generere feilrapporter om Perl, eller modulene som kommer med den, og sender dem
perldoc	Viser et stykke dokumentasjons i pod format som er innebygd i Perl installasjonstreet eller i et Perl skript
perlivp	Perl verifiseringsprosedyre for installasjonen; det kan brukes til bekrefte at Perl og dets biblioteker er installert riktig
perlthanks	Brukes til å generere takkemeldinger på E-post til Perl utviklere
piconv	En Perl versjon av tegnkodingskonverteren **iconv**
pl2pm	Et grovt verktøy for å konvertere Perl4 `.pl` filer til Perl5 `.pm` moduler
pod2html	Konverterer filer fra pod format til HTML format
pod2man	Konverterer pod data til formatert *roff inndata

pod2text	Konverterer pod data til formatert ASCII tekst
pod2usage	Skriver ut bruksmeldinger fra innebygde pod dokumenter i filer
podchecker	Kontrollerer syntaksen til dokumentasjonsfiler i podformat
podselect	Viser valgte deler av poddokumentasjonen
prove	Kommandolinjeverktøy for å kjøre tester mot Test::Harness moduler
ptar	Et **tar** likt program skrevet i Perl
ptardiff	Et Perl program som sammenligner et ekstrahert arkiv med et uekstrahert
ptargrep	Et Perl program som bruker mønstertilpasning på innholdet av filer i et tararkiv
shasum	Skriver ut eller kontrollerer SHA sjekksummer
splain	Brukes til å fremtvinge detaljert advarselsdiagnostikk i Perl
xsubpp	Konverterer Perl XS-kode til C-kode
zipdetails	Viser detaljer om den interne strukturen til en Zip-fil

8.42. XML::Parser-2.46

XML::Parser modulen er et Perl grensesnitt til James Clarks XML-parser, Expat.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 2.3 MB

8.42.1. Installasjon av XML::Parser

Forbered XML::Parser for kompilering:

```
perl Makefile.PL
```

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make test
```

Installer pakken:

```
make install
```

8.42.2. Innhold i XML::Parser

Installert modul: Expat.so

Korte beskrivelser

`Expat` gir Perl Expat grensesnittet

8.43. Intltool-0.51.0

Intltool er et internasjonaliseringsverktøy som brukes til å trekke ut oversettbare strenger fra kildefiler.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 1.5 MB

8.43.1. Installasjon av Intltool

Rett først en advarsel som er forårsaket av perl-5.22 og senere:

```
sed -i 's:\\\${:\\\$\\{:' intltool-update.in
```

Note

Det regulære uttrykket ovenfor ser uvanlig ut på grunn av alle skråstreker. Det den gjør er å legge til et skråstrek før høyre krøllparentes i sekvensen '\${' som resulterer i '\$\{'.

Forbered Intltool for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
install -v -Dm644 doc/I18N-HOWTO /usr/share/doc/intltool-0.51.0/I18N-HOWTO
```

8.43.2. Innhold i Intltool

Installerte programmer: intltool-extract, intltool-merge, intltool-prepare, intltool-update, og intltoolize
Installerte mapper: /usr/share/doc/intltool-0.51.0 og /usr/share/intltool

Korte beskrivelser

intltoolize Forbereder en pakke for å bruke intltool

intltool-extract Genererer deklarasjonsfiler som kan leses av **gettext**

intltool-merge Slår sammen oversatte strenger til forskjellige filtyper

intltool-prepare Oppdaterer pot filer og slår dem sammen med oversettelsesfiler

intltool-update Oppdaterer po malfilene og slår dem sammen med oversettelsene

8.44. Autoconf-2.71

Autoconf pakken inneholder programmer for å produsere skallskript som automatisk kan konfigurere kildekoden.

Omtrentlig byggetid: mindre enn 0.1 SBU (omtrent 6.2 SBU med tester)
Nødvendig diskplass: 24 MB

8.44.1. Installasjon av Autoconf

Først, fiks flere problemer med testene forårsaket av bash-5.2 og senere:

```
sed -e 's/SECONDS|/&SHLVL|/'               \
    -e '/BASH_ARGV=/a\        /^SHLVL=/ d' \
    -i.orig tests/local.at
```

Forbered Autoconf for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Note

Testtiden for autoconf kan reduseres betydelig på en system med flere kjerner. For å gjøre dette, legg til **TESTSUITEFLAGS=-j<N>** til linjen over. For eksempel ved å bruke -j4 kan testtiden reduseres med over 60 prosent.

Installer pakken:

```
make install
```

8.44.2. Innhold i Autoconf

Installerte programmer: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, og ifnames
Installert mappe: /usr/share/autoconf

Korte beskrivelser

autoconf Produserer skallskript som automatisk konfigurerer programvares kildekodepakker for å tilpasse seg mange typer Unix-lignende systemer; konfigurasjonsskriptene den produserer er uavhengige—å kjøre de krever ikke **autoconf** programmet

autoheader Et verktøy for å lage malfiler av C *#define* uttrykk for configure å bruke

autom4te En innpakning for M4 makroprosessoren

autoreconf Kjører automatisk **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize**, og **libtoolize** i riktig rekkefølge for å spare tid når det gjøres endringer i **autoconf** og **automake** malfiler

autoscan Hjelper med å lage en `configure.in` fil for en programvarepakke; den undersøker kildefilene i et mappetre, søker etter vanlige problemer med portabilitet, og oppretter en `configure.scan` fil som fungerer som en innledende `configure.in` fil for en pakke

autoupdate Endrer en `configure.in` fil som fortsatt anroper **autoconf** makroer ved deres gamle navn til å bruke gjeldende makronavn

ifnames Hjelper når det skrives `configure.in` 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 `configure.in` fil generert av **autoscan**.]

8.45. Automake-1.16.5

Automake pakken inneholder programmer for å generere Make filer for bruk med Autoconf.

Omtrentlig byggetid: mindre enn 0.1 SBU (omtrent 7.3 SBU med tester)
Nødvendig diskplass: 114 MB

8.45.1. Installasjon av Automake

Forbered Automake for kompilering:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.16.5
```

Kompiler pakken:

```
make
```

Å bruke make -j4 alternativet øker hastigheten på testene, selv på systemer med kun én prosessor, på grunn av interne forsinkelser i individuelle tester. Å teste resultatene, utsted:

```
make -j4 check
```

Testen t/subobj.sh er kjent for å mislykkes.

Installer pakken:

```
make install
```

8.45.2. Innholdet i Automake

Installerte programmer: aclocal, aclocal-1.16 (hardlinket til aclocal), automake, og automake-1.16 (hardlinket til automake)
Installerte mapper: /usr/share/aclocal-1.16, /usr/share/automake-1.16, og /usr/share/doc/automake-1.16.5

Korte beskrivelser

aclocal Genererer `aclocal.m4` filer basert på innholdet i `configure.in` filene

aclocal-1.16 En hard lenke til **aclocal**

automake Et verktøy for automatisk generering av `Makefile.in` filer fra `Makefile.am` filer [For å lage alle `Makefile.in` filer for en pakke, kjør dette programmet i mappen på øverste nivå. Ved å skanne `configure.in` filen, finner den automatisk hver passende `Makefile.am` fil og genererer tilsvarende `Makefile.in` fil.]

automake-1.16 En hard lenke til **automake**

8.46. OpenSSL-3.0.8

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: 3.2 SBU
Nødvendig diskplass: 520 MB

8.46.1. Installasjon av OpenSSL

Forbered OpenSSL for kompilering:

```
./config --prefix=/usr         \
         --openssldir=/etc/ssl \
         --libdir=lib          \
         shared                \
         zlib-dynamic
```

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make test
```

En test, 30-test_afalg.t, er kjent for å mislykkes på noen kjernekonfigurasjoner (avhengig av inkonsistente verdier for CONFIG_CRYPTO_USER_API*-innstillinger.) Hvis den mislykkes, kan den trygt bli ignorert.

Installer pakken:

```
sed -i '/INSTALL_LIBS/s/libcrypto.a libssl.a//' Makefile
make MANSUFFIX=ssl install
```

Legg til versjonen i dokumentasjonskatalognavnet, for å være i samsvarer med andre pakker:

```
mv -v /usr/share/doc/openssl /usr/share/doc/openssl-3.0.8
```

Hvis ønskelig, installer litt tilleggsdokumentasjon:

```
cp -vfr doc/* /usr/share/doc/openssl-3.0.8
```

Note

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 rekompilere 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 Section 8.2.1, “Oppgraderingsproblemer” for detaljer.

8.46.2. Innhold i OpenSSL

Installerte programmer: c_rehash og openssl
Installerte biblioteker: libcrypto.so og libssl.so
Installerte mapper: /etc/ssl, /usr/include/openssl, /usr/lib/engines og /usr/share/doc/openssl-3.0.8

Korte beskrivelser

c_rehash er et Perl skript som skanner alle filer i en katalog og legger til symbolske lenker til deres hash-verdier. 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 **man 1 openssl**

`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, dokumentasjon kan bli funnet ved å kjøre **man 7 ssl**

8.47. Kmod-30

Kmod pakken inneholder biblioteker og verktøy for lasting av kjernemoduler

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 12 MB

8.47.1. Installasjon av Kmod

Forbered Kmod for kompilering:

```
./configure --prefix=/usr          \
            --sysconfdir=/etc      \
            --with-openssl         \
            --with-xz              \
            --with-zstd            \
            --with-zlib
```

Betydningen av konfigureringsalternativene:

`--with-openssl`
Dette alternativet gjør det mulig for Kmod å håndtere PKCS7 signaturer for kjernemoduler.

`--with-xz`, *`--with-zlib`*, og *`--with-zstd`*
Disse alternativene gjør at Kmod kan håndtere komprimerte kjernemoduler.

Kompiler pakken:

```
make
```

Testpakken til denne pakken krever rå kjernedeklarasjoner (ikke de “sanitiserte” kjernedeklarasonene installert tidligere), som er utenfor rammen av LFS.

Installer pakken og lag symbolkoblinger for kompatibilitet med Module-Init-Tools (pakken som tidligere håndterte Linux kjernemoduler):

```
make install

for target in depmod insmod modinfo modprobe rmmod; do
  ln -sfv ../bin/kmod /usr/sbin/$target
done

ln -sfv kmod /usr/bin/lsmod
```

8.47.2. Innhold i Kmod

Installerte programmer: depmod (lenker til kmod), insmod (lenker til kmod), kmod, lsmod (lenker til kmod), modinfo (lenker til kmod), modprobe (lenker til kmod), og rmmod (lenker til kmod)
Installert bibliotek: libkmod.so

Korte beskrivelser

depmod Oppretter en avhengighetsfil basert på symbolene den finner i eksisterende sett med moduler; denne avhengighetsfilen brukes av **modprobe** for automatisk å laste de nødvendige 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

8.48. Libelf fra Elfutils-0.188

Libelf er et bibliotek for håndtering av ELF (kjørbare og linkbare formater) filer.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 120 MB

8.48.1. Installasjon av Libelf

Libelf er en del av elfutils-0.188 pakken. Bruk elfutils-0.188.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
```

Testen som heter `run-native-test.sh` er kjent for å feile.

Installer kun Libelf:

```
make -C libelf install
install -vm644 config/libelf.pc /usr/lib/pkgconfig
rm /usr/lib/libelf.a
```

8.48.2. Innhold i Libelf

Installert bibliotek: libelf.so (symlenke) og libelf-0.188.so
Installert mappe: /usr/include/elfutils

Korte beskrivelser

`libelf` Inneholder API funksjoner for å håndtere ELF objektfiler

8.49. Libffi-3.4.4

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: 1.8 SBU
Nødvendig diskplass: 11 MB

8.49.1. Installasjon av Libffi

Note

I likhet med GMP bygges libffi med spesifikke optimaliseringer til prosessoren som er i bruk. Hvis du bygger for et annet system, 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
```

8.49.2. Innhold i Libffi

Installert bibliotek: libffi.so

Korte beskrivelser

`libffi` Inneholder API funksjonene for fremmede funksjonsgrensesnitt

8.50. Python-3.11.2

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: 2.0 SBU
Nødvendig diskplass: 372 MB

8.50.1. Installasjon av Python 3

Forbered Python for kompilering:

```
./configure --prefix=/usr        \
            --enable-shared      \
            --with-system-expat  \
            --with-system-ffi    \
            --enable-optimizations
```

Betydningen av konfigureringsalternativene:

`--with-system-expat`
Denne bryteren muliggjør kobling mot systemversjonen av Expat.

`--with-system-ffi`
Denne bryteren muliggjør kobling mot systemversjonen av `libffi.so`.

`--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
```

Det anbefales ikke å kjøre testene på dette tidspunktet. Tester er kjent for å henge på ubestemt tid i det delvise LFS miljøet. Om ønskelig kan testene kjøres på nytt på slutten av dette kapittelet eller når Python 3 er reinstallert i BLFS. For å kjøre testene uansett, utsted **make test**.

Installer pakken:

```
make install
```

Vi bruker **pip3** kommandoen til å installer 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 det 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
```

Important

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-3.11.2/html

tar --strip-components=1  \
    --no-same-owner       \
    --no-same-permissions \
    -C /usr/share/doc/python-3.11.2/html \
    -xvf ../python-3.11.2-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 oppstrøms skaperens verdier.

8.50.2. Innhold i Python 3

Installerte programmer:	2to3, idle3, pip3, pydoc3, python3, og python3-config
Installert bibliotek:	libpython3.11.so and libpython3.so
Installerte mapper:	/usr/include/python3.11, /usr/lib/python3, og /usr/share/doc/python-3.11.2

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

8.51. Wheel-0.38.4

Wheel er et Python bibliotek som er referanseimplementeringen av Python wheel pakkestandarden.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 1.3 MB

8.51.1. Installasjon av Wheel

Kompiler Wheel med følgende kommando:

```
PYTHONPATH=src pip3 wheel -w dist --no-build-isolation --no-deps $PWD
```

Installer wheel med følgende kommando:

```
pip3 install --no-index --find-links=dist wheel
```

Betydningen av pip3 konfigurasjonsalternativene og kommandoene:

`PYTHONPATH=src`
Lar denne pakken (ikke installert ennå) å bygge et wheelarkiv for seg selv, for å unngå et kylling-eller-egg-problem.

wheel
Denne kommandoen bygger wheelarkivet for denne pakken.

`-w dist`
Instruerer pip å sette det opprettede wheel i `dist` mappen.

install
Denne kommandoen installerer pakken.

`--no-build-isolation`, *`--no-deps`*, og *`--no-index`*
Hindrer pip fra å hente filer fra pakkens nettdepot (PyPI). Hvis pakkene er installert i riktig rekkefølge, da trenger den ikke å hente noen filer med det første, men dette alternativet gir en viss sikkerhet i tilfelle brukerfeil.

`--find-links dist`
Instruerer pip til å søke etter wheelarkiver i `dist` mappen.

8.51.2. Innholdet i Wheel

Installert program: wheel
Installerte mapper: /usr/lib/python3.11/site-packages/wheel og /usr/lib/python3.11/site-packages/wheel-0.38.4.dist-info

Kort beskrivelse

wheel er et verktøy for å pakke ut, pakke eller konvertere wheelarkiver

8.52. Ninja-1.11.1

Ninja er et lite byggesystem med fokus på hastighet.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 77 MB

Tip

Denne delen er strengt tatt ikke nødvendig for LFS hvis det ikke brukes systemd. På den annen side, ninja knyttet til meson gir et kraftig byggesystemkombinasjon, som forventes å bli brukt stadig oftere. Det kreves for flere pakker i *BLFS boken*.

8.52.1. 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
```

Betydningen av byggealternativet:

`--bootstrap`
Denne parameteren tvinger ninja til å gjenoppbygge seg selv for gjeldene system.

For å teste resultatene, utsted:

```
./ninja ninja_test
./ninja_test --gtest_filter=-SubprocessTest.SetWithLots
```

Installer pakken:

```
install -vm755 ninja /usr/bin/
install -vDm644 misc/bash-completion /usr/share/bash-completion/completions/ninja
install -vDm644 misc/zsh-completion  /usr/share/zsh/site-functions/_ninja
```

8.52.2. Innhold av Ninja

Installerte programmer: ninja

Korte beskrivelser

ninja er Ninja byggesystemet

8.53. Meson-1.0.0

Meson er et åpen kildekode byggesystem ment å være både ekstremt raskt og så brukervennlig som mulig.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 41 MB

Tip

Denne delen er strengt tatt ikke nødvendig når LFS ikke bruker systemd. På den annen side gjør Meson, sammen med Ninja, en kraftig byggesystemkombinasjon, som nok vil bli brukt stadig oftere. Det kreves for flere pakker i *BLFS boken*.

8.53.1. Installasjon av Meson

Kompiler Meson med følgende kommando:

```
pip3 wheel -w dist --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.

8.53.2. Innhold i Meson

Installerte programmer: meson
Installert mappe: /usr/lib/python3.11/site-packages/meson-1.0.0.dist-info og /usr/lib/python3.11/site-packages/mesonbuild

Korte beskrivelser

meson Et byggesystem med høy produktivitet

8.54. Coreutils-9.1

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

Omtrentlig byggetid: 0.9 SBU
Nødvendig diskplass: 156 MB

8.54.1. 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-9.1-i18n-1.patch
```

Note

Tidligere ble det funnet mange feil i denne oppdateringen. Ved melding om nye feil til Coreutils vedlikeholdere, vennligst først sjekk om de er reproduserbare uten denne oppdateringen.

Forbered nå Coreutils for kompilering:

```
autoreconf -fiv
FORCE_UNSAFE_CONFIGURE=1 ./configure \
            --prefix=/usr            \
            --enable-no-install-program=kill,uptime
```

Betydningen av konfigureringsalternativene:

autoreconf
Oppdateringen for internasjonalisering har modifisert byggesystemet til pakken, slik at konfigurasjonsfilene må bli regenerert.

`FORCE_UNSAFE_CONFIGURE=1`
Denne miljøvariabelen lar pakken bli bygget som `root` brukeren.

`--enable-no-install-program=kill,uptime`
Hensikten med denne bryteren er å hindre Coreutils fra 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 én gruppe. Sånn at disse testene ikke hoppes over, legg til en midlertidig gruppe og gjør bruker `tester` en del av de:

```
echo "dummy:x:102:tester" >> /etc/group
```

Fiks noen av tillatelsene slik at ikke-`root` brukeren kan kompilere og kjøre testene:

```
chown -Rv tester .
```

Kjør nå testene:

```
su tester -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes check"
```

test-getlogin testen kan mislykkes i LFS chroot miljøet.

Fjern den midlertidige gruppen:

```
sed -i '/dummy/d' /etc/group
```

Installer pakken:

```
make install
```

Flytt programmer til stedene spesifisert av FHS:

```
mv -v /usr/bin/chroot /usr/sbin
mv -v /usr/share/man/man1/chroot.1 /usr/share/man/man8/chroot.8
sed -i 's/"1"/"8"/' /usr/share/man/man8/chroot.8
```

8.54.2. Innhold i Coreutils

Installerte programmer: [, b2sum, base32, base64, basename, basenc, cat, chcon, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, numfmt, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami, og yes
Installert bibliotek: libstdbuf.so (i /usr/libexec/coreutils)
Installert mappe: /usr/libexec/coreutils

Korte beskrivelser

[Er faktisk en kommando, /usr/bin/[; det er et synonym for **test** kommandoen

base32 Koder og dekoder data i henhold til base32 spesifikasjonen (RFC 4648)

base64 Koder og dekoder data i henhold til base64 spesifikasjonen (RFC 4648)

b2sum Skriver ut eller kontrollerer BLAKE2 (512-bit) sjekksummer

basename Fjerner enhver bane og et gitt suffiks fra et filnavn

basenc Koder eller dekoder data ved hjelp av ulike algoritmer

cat Slår sammen filer til standard 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 sjekksummen for syklisk redundanssjekk (CRC) og antall byte for hver spesifisert fil

comm Sammenligner to sorterte filer, og skriver ut i tre kolonner, linjene som er unike og linjene som er vanlige

cp Kopierer filer

csplit Deler en gitt fil i flere nye filer, og skiller dem i henhold til gitte mønstre eller linjenummer og skriver ut antall byte av hver nye fil

cut Skriver ut seksjoner av linjer, og velger delene i henhold til gitte felt eller posisjoner

date Viser gjeldende 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 `LS_COLOR` miljøvariabelen for å endre fargeskjemaet som brukes av **ls**

dirname Trekker ut mappedelen(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 tabulatortegn

pathchk Sjekker om filnavn er gyldige eller flyttbare

pinky Er en lettvekts fingerklient; den rapporterer noe informasjon om de gitte brukerne

pr Paginerer og spalter filer for utskrift

printenv Skriver ut miljøet

printf Skriver ut de gitte argumentene i henhold til det gitte formatet, mye som C printf funksjonen

ptx Produserer en permutert indeks fra innholdet i de gitte filene, med hvert søkeord i sin kontekst

pwd Rapporterer navnet på gjeldende 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 blokktellinger for hver gitt fil

sync Tømmer filsystembuffere; den tvinger endrede blokker til disk og oppdaterer superblokken

tac Sammenslår de gitte filene i revers

tail Skriver ut de ti siste linjene (eller gitt antall linjer) av hver gitt fil

tee Leser fra standard inngang mens du skriver både til standard utgang og til de gitte filene

test Sammenligner verdier og kontrollerer filtyper

timeout Kjører en kommando med en tidsbegrensning

touch Endrer filtidsstempler, angir tilgang og 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 én fil er gitt

who Rapporterer hvem som er pålogget

whoami Rapporterer brukernavnet som er knyttet til gjeldende effektive bruker-ID

yes Skriver ut “y”, gjentatte ganger eller en gitt streng til den drepes

`libstdbuf` Bibliotek brukt av **stdbuf**

8.55. Check-0.15.2

Check er et rammeverk for C enhetstesting.

Omtrentlig byggetid: 0.1 SBU (omtrent 1.7 SBU med tester)
Nødvendig diskplass: 12 MB

8.55.1. 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
```

8.55.2. 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

8.56. Diffutils-3.9

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

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 35 MB

8.56.1. Installasjon av Diffutils

Forbered Diffutils for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.56.2. 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

8.57. Gawk-5.2.1

Gawk pakken inneholder programmer for å manipulere tekstfiler.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 43 MB

8.57.1. Installasjon av Gawk

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

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

Forbered Gawk for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make LN='ln -f' install
```

Betydningen av den overstyrte make variabelen:

`LN='ln -f'`

Denne variabelen sikrer at den forrige harde lenken installert i Section 6.9, “Gawk-5.2.1” er oppdatert her.

Hvis ønskelig, installer dokumentasjonen:

```
mkdir -pv                                   /usr/share/doc/gawk-5.2.1
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-5.2.1
```

8.57.2. Innhold i Gawk

Installerte programmer: awk (lenker til gawk), gawk, og gawk-5.2.1
Installerte biblioteker: filefuncs.so, fnmatch.so, fork.so, inplace.so, intdiv.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoway.so, rwarray.so, og time.so (alle i /usr/lib/gawk)
Installerte mapper: /usr/lib/gawk, /usr/libexec/awk, /usr/share/awk, og /usr/share/doc/gawk-5.2.1

Korte beskrivelser

awk En lenke til **gawk**

gawk Et program for å manipulere tekstfiler; det er GNU implementeringen av **awk**

gawk-5.2.1 En hard lenke til **gawk**

8.58. Findutils-4.9.0

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: 0.4 SBU
Nødvendig diskplass: 51 MB

8.58.1. Installasjon av Findutils

Forbered Findutils for kompilering:

```
case $(uname -m) in
    i?86)   TIME_T_32_BIT_OK=yes ./configure --prefix=/usr --localstatedir=/var/lib/locate ;;
    x86_64) ./configure --prefix=/usr --localstatedir=/var/lib/locate ;;
esac
```

Betydningen av konfigureringsalternativene:

TIME_32_BIT_OK=yes
Denne innstillingen er nødvendig for å bygge et 32 bit system.

`--localstatedir`
Dette alternativet flytter **locate** databasen til `/var/lib/locate`, som er den FHS kompatible plasseringen.

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

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

Installer pakken:

```
make install
```

8.58.2. 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

8.59. Groff-1.22.4

Groff pakken inneholder programmer for prosessering og formatering av tekst.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 89 MB

8.59.1. 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
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make install
```

8.59.2. Innhold i Groff

Installerte programmer: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, glilypond, gperl, gpinyin, grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, gropdf, grops, grotty, hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfmom, pdfroff, pfbtops, pic, pic2graph, post-grohtml, preconv, pre-grohtml, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit, og troff
Installerte mapper: /usr/lib/groff og /usr/share/doc/groff-1.22.4, /usr/share/groff

Korte beskrivelser

addftinfo Leser en troff fontfil og legger til noen ekstra fontmetrikk informasjon som brukes av **groff** systemet

afmtodit Oppretter en fontfil for bruk med **groff** og **grops**

chem Groff 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 dokumentformateringssystemet; 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 `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s`, og `-t` 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 `.pfb` formatet til ASCII
pic	Kompilerer beskrivelser av bilder innebygd i troff eller TeX inndatafiler til kommandoer som forstås av TeX eller **troff**
pic2graph	Konverterer et PIC diagram til et beskåret bilde
post-grohtml	Oversetter utdataene 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 *.[* og *.]* som tolkes som referanser, og linjer mellom *.R1* og *.R2* 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 *.so file* av innholdet i nevnte *file*
tbl	Kompilerer beskrivelser av tabeller innebygd i troff inndatafiler til kommandoer som forstås av **troff**
tfmtodit	Oppretter en fontfil for bruk med **groff -Tdvi**
troff	Er veldig kompatibel med Unix **troff**; den bør vanligvis startes ved hjelp av **groff** kommandoen, som også vil kjøre forbehandler og etterbehandler i riktig rekkefølge og med passende alternativer

8.60. GRUB-2.06

GRUB pakken inneholder en oppstartslaster (GRand Unified Bootloader).

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 161 MB

8.60.1. Installasjon av GRUB

Note

Hvis systemet ditt har UEFI støtte og du ønsker å starte LFS med UEFI, kan du hoppe over denne pakken i LFS, og installere GRUB med UEFI støtte (og dets avhengigheter) ved å følge instruksjonene på *BLFS siden*.

Warning

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.

Løs et problem som forårsaker at **grub-install** mislykkes når `/boot` partisjon (eller root partisjon hvis `/boot` ikke er en separat partisjon) er opprettet av e2fsprogs-1.47.0 eller nyere:

```
patch -Np1 -i ../grub-2.06-upstream_fixes-1.patch
```

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 begrensete LFS miljøet. For å kjøre testene uansett, kjør **make check**.

Installer pakken:

```
make install
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

Å gjøre LFS systemet oppstartbart med GRUB vil bli diskutert i Section 10.4, “Bruke GRUB til å sette opp oppstartsprosessen”.

8.60.2. Innhold i GRUB

Installerte programmer: grub-bios-setup, grub-editenv, grub-file, grub-fstest, grub-glue-efi, grub-install, grub-kbdcomp, grub-macbless, grub-menulst2cfg, grub-mkconfig, grub-mkimage, grub-mklayout, grub-mknetdir, grub-mkpasswd-pbkdf2, grub-mkrelpath, grub-mkrescue, grub-mkstandalone, grub-ofpathname, grub-probe, grub-reboot, grub-render-label, grub-script-check, grub-set-default, grub-sparc64-setup, og grub-syslinux2cfg

Installerte mapper: /usr/lib/grub, /etc/grub.d, /usr/share/grub, og /boot/grub (når grub-install kjøres for første gang)

Korte beskrivelser

grub-bios-setup Er et hjelpeprogram for **grub-install**

grub-editenv Et verktøy for å redigere miljøblokken

grub-file Sjekker om den gitte filen er av den angitte typen

grub-fstest Verktøy for å feilsøke filsystemdriveren

grub-glue-efi Limer 32-biters og 64-biters binære filer til én 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-menulst2cfg Konverterer en GRUB Legacy `menu.lst` til en `grub.cfg` for bruk med GRUB 2

grub-mkconfig Generer en `grub.cfg` fil

grub-mkimage Lager et oppstartbart bilde av GRUB

grub-mklayout Genererer en GRUB tastaturoppsettfil

grub-mknetdir Forbereder en GRUB netboot mappe

grub-mkpasswd-pbkdf2 Genererer et kryptert PBKDF2 passord for bruk i oppstartsmenyen

grub-mkrelpath Gir et systembanenavn i forhold til roten

grub-mkrescue Lager et oppstartbart bilde av GRUB som passer for en diskett, 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 .disk_label 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

8.61. Gzip-1.12

Gzip pakken inneholder programmer for komprimering og dekomprimering av filer.

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 21 MB

8.61.1. Installasjon av Gzip

Forbered Gzip for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.61.2. Contents of Gzip

Installerte programmer: gunzip, gzexe, gzip, uncompress (hard lenket med gunzip), zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore, og znew

Korte beskrivelser

gunzip Dekomprimerer gzippede filer

gzexe Oppretter selvdekomprimerende kjørbare filer

gzip Komprimerer de gitte filene ved å bruke Lempel-Ziv (LZ77) koding

uncompress Dekomprimerer komprimerte filer

zcat Dekomprimerer de gitte gzip filene til standard utgang

zcmp Kjører **cmp** på gzippede filer

zdiff Kjører **diff** på gzippede filer

zegrep Kjører **egrep** på gzippede filer

zfgrep Kjører **fgrep** på gzippede filer

zforce Tvinger et `.gz` filetternavn på alle gitte filer som er gzippede filer, slik at **gzip** ikke vil komprimere dem igjen; dette kan være nyttig når filnavn ble avkortet under en filoverføring

zgrep Kjører **grep** på gzippede filer

zless Kjører **less** på gzippede filer

zmore Kjører **more** på gzippede filer

znew Re-komprimerer filer fra **compress** format til **gzip** format—`.Z` til `.gz`

8.62. IPRoute2-6.1.0

IPRoute2 pakken inneholder programmer for grunnleggende og avansert IPV4 basert nettverksbygging.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 17 MB

8.62.1. 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. (Hvis **arpd** programmet er nødvendig, instruksjoner for kompilering av Berkeley DB finnes i BLFS boken på *https://www.linuxfromscratch.org/blfs/view/11.3/server/db.html*.)

```
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:

```
mkdir -pv             /usr/share/doc/iproute2-6.1.0
cp -v COPYING README* /usr/share/doc/iproute2-6.1.0
```

8.62.2. Innhold i IPRoute2

Installerte programmer: bridge, ctstat (lenker til lnstat), genl, ifstat, ip, lnstat, nstat, routel, rtacct, rtmon, rtpr, rtstat (lenker til lnstat), ss, og tc
Installerte mapper: /etc/iproute2, /usr/lib/tc, og /usr/share/doc/iproute2-6.1.0

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 rutingtabellen og endre rutetabellregler
ip tunnel lar brukere se på IP tunneler og deres egenskaper, og endre dem

ip maddr lar brukerne se på multicast adresser og deres egenskaper, og endre dem
ip mroute lar brukere angi, endre eller slette multicast rutingen
ip monitor lar brukerne overvåke kontinuerlig tilstanden til enheter, adresser og ruter

lnstat Gir Linux nettverksstatistikk; det er en generalisert og mer funksjonsfull erstatning for det gamle **rtstat** programmet

nstat Viser nettverksstatistikk

routel En komponent av **ip route**, for å liste rutetabellene

rtacct Viser innholdet i `/proc/net/rt_acct`

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.

8.63. Kbd-2.5.1

Kbd pakken inneholder tastaturfiler, konsollfonter og tastaturverktøy.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 35 MB

8.63.1. Installasjon av Kbd

Oppførselen til tilbaketastene og slettetastene er ikke konsistent på tvers av tastaturoppsettene i Kbd pakken. Følgende oppdatering fikser dette problemet for i386 tastaturoppsett:

```
patch -Np1 -i ../kbd-2.5.1-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
```

Note

For noen språk (f.eks. hviterussisk) gir ikke Kbd pakken et nyttig tastaturoppsett hvor beholdningen “av” tastaturoppsettene antar en ISO-8859-5-koding og CP1251 tastaturet vanligvis brukes. Brukere av slike språk må laste ned et fungerende tastaturoppsett separat.

Hvis ønskelig, installer dokumentasjonen:

```
mkdir -pv           /usr/share/doc/kbd-2.5.1
cp -R -v docs/doc/* /usr/share/doc/kbd-2.5.1
```

8.63.2. Innhold i Kbd

Installerte programmer: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (lenker til psfxtable), psfgettable (lenker til psfxtable), psfstriptable (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-2.5.1, 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
psfstriptable	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

8.64. Libpipeline-1.5.7

Libpipeline pakken inneholder et bibliotek for å manipulere kanaler av delprosesser på en fleksibel og praktisk måte.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 10 MB

8.64.1. Installasjon av Libpipeline

Forbered Libpipeline for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.64.2. Innhold i Libpipeline

Installert bibliotek: libpipeline.so

Korte beskrivelser

`libpipeline` Dette biblioteket brukes til å trygt konstruere kanaler mellom delprosesser

8.65. Make-4.4

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

Omtrentlig byggetid: 0.5 SBU
Nødvendig diskplass: 13 MB

8.65.1. Installasjon av Make

Først, fiks noen problemer identifisert oppstrøms:

```
sed -e '/ifdef SIGPIPE/,+2 d' \
    -e '/undef  FATAL_SIG/i FATAL_SIG (SIGPIPE);' \
    -i src/main.c
```

Forbered Make for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.65.2. 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

8.66. 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

8.66.1. Installasjon av patch

Forbered patch for kompilering:

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

Kompiler pakken:

```
make
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

8.66.2. Innhold i oppdateringen

Installert program: patch

Korte beskrivelser

patch Endrer filer i henhold til en patch fil (En patch fil er normalt en forskjellsoppføring opprettet med **diff** programmet. Ved å bruke disse forskjellene på originalfilene, **patch** oppretter de lappede versjonene.)

8.67. Tar-1.34

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: 1.5 SBU
Nødvendig diskplass: 40 MB

8.67.1. 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 på filsystemet som brukes til å bygge LFS.

Installer pakken:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.34
```

8.67.2. Innhold i Tar

Installerte programmer: tar
Installert mappe: /usr/share/doc/tar-1.34

Korte beskrivelser

tar Oppretter, trekker ut filer fra og viser innholdet i arkiver, også kjent som tarballer

8.68. Texinfo-7.0.2

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

Omtrentlig byggetid: 0.3 SBU
Nødvendig diskplass: 128 MB

8.68.1. 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 treet hvis for eksempel en TeX pakke vil bli installert senere.

Infodokumentasjonssystemet bruker en ren tekstfil til å holde listen over menyoppføringer. Filen ligger på `/usr/share/info/dir`. Dessverre, på grunn av sporadiske problemer i Makefiles for forskjellige pakker, kan det noen ganger gå ut av synkronisering med infosidene som er installert på systemet. Hvis `/usr/share/info/dir` filen noen gang trenger å bli gjenskapt, vil følgende valgfrie kommandoer utføre oppgaven:

```
pushd /usr/share/info
  rm -v dir
  for f in *
    do install-info $f dir 2>/dev/null
  done
popd
```

8.68.2. 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

8.69. Vim-9.0.1273

Vim pakken inneholder en kraftig tekstredigerer.

Omtrentlig byggetid: 2.4 SBU
Nødvendig diskplass: 235 MB

Alternativer til Vim

Hvis du foretrekker en annen tekstredigerer—som Emacs, Joe, eller Nano—Vennligst se *https://www.linuxfromscratch.org/blfs/view/11.3/postlfs/editors.html* for foreslåtte installasjonsinstruksjoner.

8.69.1. Installasjon av Vim

Først endrer du standardplasseringen for `vimrc` konfigurasjonsfil til `/etc`:

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Forbered vim for kompilering:

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

Kompiler pakken:

```
make
```

For å forberede testene, sørg for at brukeren `tester` kan skrive til kildetreet:

```
chown -Rv tester .
```

Kjør nå testene som bruker `tester`:

```
su tester -c "LANG=en_US.UTF-8 make -j1 test" &> vim-test.log
```

Testpakken sender ut mange binære data til skjermen. Dette kan forårsake problemer med innstillingene til gjeldende terminal. Problemet kan unngås ved å omdirigere utdataene til en loggfil som vist ovenfor. En vellykket test vil resultere i ordene "ALL DONE" i loggfilen ved ferdigstillelse.

Installer pakken:

```
make install
```

Mange brukere 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 mansiden 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-9.0.1273`, som gjør det i samsvar med plasseringen av dokumentasjonen for andre pakker:

```
ln -sv ../vim/vim90/doc /usr/share/doc/vim-9.0.1273
```

Hvis et X Window System skal installeres på LFS systemet, kan det være nødvendig å rekompilere vim etter installasjon av X. Vim kommer med en GUI versjon av tekstredigereren som krever X og noen flere biblioteker som skal installeres. For mer informasjon om denne prosessen, se vim dokumentasjonen og vim installasjonssiden i BLFS boka på *https://www.linuxfromscratch.org/blfs/view/11.3/postlfs/vim.html*.

8.69.2. 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'
```

Note

Som standard installerer vim kun stavefiler for det engelske språket. For å installere stavefiler for ditt foretrukne språk, kopier `.spl` og eventuelt `.sug` filer for ditt språk og tegnkoding fra `runtime/spell` til mappen `/usr/share/vim/vim90/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`.

8.69.3. 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

8.70. Eudev-3.2.11

Eudev pakken inneholder programmer for dynamisk oppretting av enhetsnoder.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 83 MB

8.70.1. Installasjon av Eudev

Først fiks plasseringen av udev regler i .pc filen:

```
sed -i '/udevdir/a udev_dir=${udevdir}' src/udev/udev.pc.in
```

Forbered Eudev for kompilering:

```
./configure --prefix=/usr           \
            --bindir=/usr/sbin      \
            --sysconfdir=/etc       \
            --enable-manpages       \
            --disable-static
```

Kompiler pakken:

```
make
```

Opprett noen mapper som er nødvendig for tester, men vil også bli brukt som en del av installasjonen:

```
mkdir -pv /usr/lib/udev/rules.d
mkdir -pv /etc/udev/rules.d
```

For å teste resultatene, utsted:

```
make check
```

Installer pakken:

```
make install
```

Installer noen tilpassede regler og støttefiler som er nyttige i et LFS miljø:

```
tar -xvf ../udev-lfs-20171102.tar.xz
make -f udev-lfs-20171102/Makefile.lfs install
```

8.70.2. Konfigurerer Eudev

Informasjon om maskinvareenheter vedlikeholdes i `/etc/udev/hwdb.d` og `/usr/lib/udev/hwdb.d` mappene. Eudev trenger denne informasjonen for å bli kompilert inn i en binær database `/etc/udev/hwdb.bin`. Opprett innledende database:

```
udevadm hwdb --update
```

Denne kommandoen må kjøres hver gang maskinvareinformasjonen blir oppdatert.

8.70.3. Innhold i Eudev

Installerte programmer: udevadm og udevd
Installerte biblioteker: libudev.so
Installerte mapper: /etc/udev, /usr/lib/udev, og /usr/share/doc/udev-udev-lfs-20171102

Korte beskrivelser

udevadm Generisk administrasjonsverktøy for udev: kontrollerer udevd nissen (daemon), gir informasjon fra Udev databasen, overvåker uevents, venter på at uevents fullføres, tester Udev konfigurasjonen og trigger uevents for en gitt enhet

udevd En nisse som lytter etter uevents på netlink kontakten, oppretter enheter og kjører de konfigurerte eksterne programmene i svar på disse hendelsene

`libudev` Et bibliotekgrensesnitt for udev enhetsinformasjon

`/etc/udev` Inneholder Udev konfigurasjonsfiler, enhetstillatelser og regler for å navngi enheter

8.71. Man-DB-2.11.2

Man-DB pakken inneholder programmer for å finne og se på mansider.

Omtrentlig byggetid: 0.2 SBU
Nødvendig diskplass: 40 MB

8.71.1. Installasjon av Man-DB

Forbered Man-DB for kompilering:

```
./configure --prefix=/usr                         \
            --docdir=/usr/share/doc/man-db-2.11.2 \
            --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 mansider. 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
```

8.71.2. Ikke-engelske manualsider i LFS

Følgende tabell viser tegnsettet som Man-DB antar manuelle sider installert under `/usr/share/man/<ll>` vil være kodet med. I tillegg til dette, bestemmer Man-DB korrekt om manualsider installert i den katalogen er UTF-8-kodet.

Table 8.1. Forventet tegnkoding av eldre 8-biters manualsider

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

Note

Manualsider på språk som ikke er på listen støttes ikke.

8.71.3. 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-2.11.2

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 manualsidene

lexgrog	Viser en-linjes sammendragsinformasjon om en gitt manualside
man	Formaterer og viser den forespurte manualsiden
man-recode	Konverterer manualsider til en annen koding
mandb	Oppretter eller oppdaterer **whatis** databasen
manpath	Viser innholdet i $MANPATH eller (hvis $MANPATH ikke er angitt) en passende søkebane basert på innstillingene i man.conf og brukerens miljø
whatis	Søker **whatis** databasen og viser de korte beskrivelsene av systemkommandoer som inneholder de gitte nøkkelord som et separat ord
`libman`	Inneholder kjøretidsstøtte for **man**
`libmandb`	Inneholder kjøretidsstøtte for **man**

8.72. Procps-ng-4.0.2

Procps-ng pakken inneholder programmer for overvåking av prosesser.

Omtrentlig byggetid: 0.1 SBU
Nødvendig diskplass: 26 MB

8.72.1. Installasjon av Procps-ng

Forbered procps-ng for kompilering:

```
./configure --prefix=/usr                           \
            --docdir=/usr/share/doc/procps-ng-4.0.2 \
            --disable-static                        \
            --disable-kill
```

Betydningen av konfigureringsalternativet:

`--disable-kill`
Denne bryteren deaktiverer bygging av **kill** kommandoen som vil bli installert av Util-linux pakken.

Kompiler pakken:

```
make
```

For å kjøre testpakken, kjør:

```
make check
```

En test kalt `free with commit` kan mislykkes hvis noen applikasjoner med en tilpasset minneallokator (for eksempel JVM og nettlesere) kjører på vertsdistroen.

Installer pakken:

```
make install
```

8.72.2. 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-4.0.2

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 det er logget på, og systemets belastningsgjennomsnitt

vmstat Rapporterer virtuelt minnestatistikk, gir informasjon om prosesser, minne, søking, blokk Input/Output (IO), feller og CPU aktivitet

w Viser hvilke brukere som for øyeblikket er pålogget, hvor og siden når

watch Kjører en gitt kommando gjentatte ganger, og viser den første skjermen full av utdata; dette lar en bruker se utdataens endring over tid

`libproc-2` Inneholder funksjonene som brukes av de fleste programmer i denne pakken

8.73. Util-linux-2.38.1

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: 283 MB

8.73.1. Installasjon av Util-linux

Forbered Util-linux for kompilering:

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --bindir=/usr/bin    \
            --libdir=/usr/lib    \
            --sbindir=/usr/sbin  \
            --disable-chfn-chsh  \
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
            --disable-runuser    \
            --disable-pylibmount \
            --disable-static     \
            --without-python     \
            --without-systemd    \
            --without-systemdsystemunitdir \
            --docdir=/usr/share/doc/util-linux-2.38.1
```

Alternativene --disable og --without forhindrer advarsler om bygningskomponenter som krever pakker som ikke er i LFS eller er inkonsistent med programmer installert av andre pakker.

Kompiler pakken:

```
make
```

Hvis ønskelig, kjør testpakken som en ikke-`root` bruker:

Warning

Å 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
```

```
chown -Rv tester .
su tester -c "make -k check"
```

hardlink testene vil mislykkes hvis vertens kjerne ikke har alternativet CONFIG_CRYPTO_USER_API_HASH satt. I tillegg, to undertester fra misc: mbsencode og en undertest fra script: replay er kjent for å mislykkes.

Installer pakken:

```
make install
```

8.73.2. Innhold i Util-linux

Installerte programmer: addpart, agetty, blkdiscard, blkid, blkzone, blockdev, cal, cfdisk, chcpu, chmem, choom, chrt, col, colcrt, colrm, column, ctrlaltdel, delpart, dmesg, eject, fallocate, fdisk, fincore, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, 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, lsfd, lslocks, lslogins, lsmem, lsns, mcookie, mesg, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, nsenter, partx, pivot_root, prlimit, readprofile, rename, renice, resizepart, rev, rfkill, rtcwake, script, scriptlive, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff, swapon, switch_root, taskset, uclampset, ul, umount, uname26 (lenker til setarch), unshare, utmpdump, uuidd, 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-2.38.1, 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

colcrt Filtrerer **nroff** utdata for terminaler som mangler noen muligheter, for eksempel overslag og halvlinjer

colrm Filtrerer ut de gitte kolonnene

column Formaterer en gitt fil i flere kolonner

ctrlaltdel Setter funksjonen til tastekombinasjonen Ctrl+Alt+Del til en hard eller myk tilbakestilling

delpart Ber Linuxkjernen om å fjerne en partisjon

dmesg Dumper kjerneoppstartsmeldingene

eject Løser ut flyttbare medier

fallocate Forhåndstildeler plass til en fil

fdisk Manipulerer partisjonstabellen til den gitte enheten

fincore Teller sider med filinnhold i kjernen

findfs Finner et filsystem etter etikett eller universell unik identifikator (UUID)

findmnt Er et kommandolinjegrensesnitt til libmount biblioteket for arbeid med mountinfo, fstab og mtab filer

flock 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 Analyserer 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 kjerneavbruddsteller i `top(1)` stilvisning

isosize Rapporterer størrelsen på et ISO9660 filsystem

kill Sender signaler til prosesser

last Viser hvilke brukere som sist logget på (og ut), søker tilbake gjennom `/var/log/wtmp` filen; den viser også systemoppstart, systemavslutning og endringer på kjørenivå

lastb Viser mislykkede påloggingsforsøk, som logget i `/var/log/btmp`

ldattach Fester en linjedisiplin til en seriellinje

linux32 En symbolsk lenke til setarch

linux64 En symbolsk lenke til setarch

logger Legger inn den gitte meldingen i systemloggen

look Viser linjer som begynner med den gitte strengen

losetup Setter opp og kontrollerer sløyfeenheter

lsblk Viser informasjon om alle eller utvalgte blokkenheter i et trelignende format

lscpu Skriver ut CPU arkitekturinformasjon

lsfd Viser informasjon om åpne filer; erstatter **lsof**

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
rkfill 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 `root` å logge inn; den er normalt startet av **init** når systemet går i enkeltbrukermodus

swaplabel 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 monteringstreet

taskset Henter eller setter en prosess sin CPU tilhørighet

uclampset Manipuler bruk av clamping attributtene til systemet eller en prosess

ul Et filter for å oversette understrek til skiftesekvenser som indikerer understreking for terminalen som er i bruk

umount Kobler et filsystem fra systemets filtre

uname26 En symbolsk lenke til setarch

unshare Kjører et program med noen navnerom som ikke er delt fra overordnet

utmpdump Viser innholdet i den gitte påloggingsfilen i et mer brukervennlig format

uuidd En nisse som brukes av UUID biblioteket for å generere tidsbasert UUID på en sikker og garantert unik måte

uuidgen Oppretter nye UUID. Hver ny UUID 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 (~340 billioner billioner billioner unike UUIDer 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 filsystemsignatur fra en enhet

x86_64 En symbolsk lenke til setarch

zramctl Et program for å sette opp og kontrollere zram (komprimert ram disk) enheter

`libblkid` Inneholder rutiner for enhetsidentifikasjon og symbol utdrag

`libfdisk` Inneholder rutiner for manipulering av partisjonstabeller

`libmount` Inneholder rutiner for montering og avmontering av en blokkenhet

`libsmartcols` Inneholder rutiner for å hjelpe skjermutdata i tabulatorform

`libuuid` Inneholder rutiner for å generere unike identifikatorer for objekter som kan være tilgjengelig utenfor det lokale systemet

8.74. E2fsprogs-1.47.0

E2fsprogs pakken inneholder verktøyene for å håndtere `ext2` filsystemet. Det støtter også `ext3` og `ext4` journalførende filsystemer.

Omtrentlig byggetid: 2.4 SBU pÃ¥ en spinnende disk, 0.4 SBU pÃ¥ en SSD
Nødvendig diskplass: 95 MB

8.74.1. Installasjon av E2fsprogs

E2fsprogs dokumentasjonen anbefaler at pakken bygges i en undermappe til kildetreet:

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

Forbered E2fsprogs for kompilering:

```
../configure --prefix=/usr           \
             --sysconfdir=/etc       \
             --enable-elf-shlibs     \
             --disable-libblkid      \
             --disable-libuuid       \
             --disable-uuidd         \
             --disable-fsck
```

Betydningen av konfigureringsalternativene:

`--enable-elf-shlibs`
Dette oppretter de delte bibliotekene som noen programmer i denne pakken bruker.

`--disable-*`
Dette hindrer E2fsprogs fra å bygge og installere `libuuid` og `libblkid` bibliotekene, `uuidd` nissen, og **fsck** innpakningen, util-linux installerer nyere versjoner.

Kompiler pakken:

```
make
```

For å kjøre testene, utsted:

```
make check
```

En test, u_direct_io, er kjent for å mislykkes på noen systemer.

Installer pakken:

```
make install
```

Fjern ubrukelige statiske biblioteker:

```
rm -fv /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Denne pakken installerer en gzipped `.info` 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
```

8.74.2. Konfigurere E2fsprogs

`/etc/mke2fs.conf` inneholder standardverdien til ulike kommandolinjealternativer for **mke2fs**. Du kan rediger filen for å gjøre standardverdiene egnet for ditt 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.

8.74.3. 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 `ext{234}` 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 `com_err` biblioteket
debugfs	En feilsøker for filsystemet; den kan brukes til å undersøke og endre tilstanden til `ext{234}` 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 `ext{234}` filsystemer
e2image	Brukes til å lagre kritiske `ext{234}` filsystemdata til en fil
e2label	Viser eller endrer filsystemetiketten på et `ext{234}` filsystem tilstede på en gitt enhet
e2mmpstatus	Sjekker MMP (Multiple Mount Protection) status for et `ext4` filsystem
e2scrub	Sjekker innholdet i et montert `ext{234}` filsystem
e2scrub_all	Sjekker alle monterte `ext{234}` filsystemer for feil
e2undo	Gjentar angreloggen (undo_log) for et `ext{234}` filsystem funnet på en enhet [Dette kan brukes til å angre en mislykket operasjon av et e2fsprogs program.]
e4crypt	`Ext4` filsystem krypteringsverktøy
e4defrag	Online defragmentering for `ext4` fil systemer
filefrag	Rapporter om hvor dårlig fragmentert en bestemt fil kan være
fsck.ext2	Som standard sjekker `ext2` filsystemer og er en hard lenke til **e2fsck**

fsck.ext3	Som standard sjekker `ext3` filsystemer og er en hard lenke til **e2fsck**
fsck.ext4	Som standard sjekker `ext4` 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 `libss` delsystembibliotek
mke2fs	Oppretter et `ext{234}` filsystem på den angitte enheten
mkfs.ext2	Som standard oppretter `ext2` filsystemer og er en hard lenke til **mke2fs**
mkfs.ext3	Som standard oppretter `ext3` filsystemer og er en hard lenke til **mke2fs**
mkfs.ext4	Som standard oppretter `ext4` filsystemer og er en hard lenke til **mke2fs**
mklost+found	Oppretter en `lost+found` mappe på et `ext{234}` filsystem ; den forhåndstildeler diskblokker til denne katalogen for å lette oppgaven til **e2fsck**
resize2fs	Kan brukes til å forstørre eller krympe et `ext{234}` filsystem
tune2fs	Justerer justerbare filsystemparametere på et `ext{234}` filsystem
`libcom_err`	Den vanlige feilvisningsrutinen
`libe2p`	Brukt av **dumpe2fs**, **chattr**, og **lsattr**
`libext2fs`	Inneholder rutiner for å gjøre det mulig for programmer på brukernivå å manipulere et `ext{234}` filsystem
`libss`	Brukt av **debugfs**

8.75. Sysklogd-1.5.1

Sysklogd pakken inneholder programmer for logging av systemmeldinger, slik som de kjernen gir når uvanlige ting skjer.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 0.7 MB

8.75.1. Installasjon av Sysklogd

Først, fiks problemer som forårsaker en segmenteringsfeil under noen forhold i klogd og fiks en foreldet programkonstruksjon:

```
sed -i '/Error loading kernel symbols/{n;n;d}' ksym_mod.c
sed -i 's/union wait/int/' syslogd.c
```

Kompiler pakken:

```
make
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make BINDIR=/sbin install
```

8.75.2. 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 *

# End /etc/syslog.conf
EOF
```

8.75.3. Innhold i Sysklogd

Installerte programmer: klogd og syslogd

Korte beskrivelser

klogd En systemnisse (system daemon) for å avskjære og logge kjernemeldinger

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.]

8.76. Sysvinit-3.06

Sysvinit pakken inneholder programmer for å kontrollere oppstarten, kjøring og avslutning av systemet.

Omtrentlig byggetid: mindre enn 0.1 SBU
Nødvendig diskplass: 4.4 MB

8.76.1. 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-3.06-consolidated-1.patch
```

Kompiler pakken:

```
make
```

Denne pakken kommer ikke med en testpakke.

Installer pakken:

```
make install
```

8.76.2. 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 kodede 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 `/var/log/wtmp` 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 `/run/utmp`

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

8.77. 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 feilsøkingsinformasjon, 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.

8.78. Stripping

Denne delen er valgfri. Hvis den tiltenkte brukeren ikke er en programmerer og ikke planlegger å gjøre noen feilsøking på systemprogramvaren, kan systemstørrelsen reduseres med omtrent 2 GB ved å fjerne feilsøkingssymbolene, 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 idé å lage en sikkerhetskopiering av LFS systemet i gjeldende tilstand.

En **strip** kommando med *--strip-unneeded* alternativet fjerner alle feilsøkingssymboler fra en binær eller et bibliotek. Og det fjerner alle symboltabelloppføringer som ikke er nødvendig for linkeren (for statiske biblioteker) eller dynamisk linker (for dynamisk koblede binære filer og delte biblioteker).

Feilsøkingssymbolene for utvalgte biblioteker er plassert i separate filer. Denne feilsøkingsinformasjonen er nødvendig hvis det kjøres regresjonstester som bruker *valgrind* eller *gdb* senere i BLFS.

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 Section 8.2.1, “Oppgraderingsproblemer” gir begrunnelsen for å bruke **install** kommandoen her.)

Note

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.

Important

Hvis en pakke som versjonen er forskjellig fra versjonen spesifisert av boken (enten etter et sikkerhetsråd eller som tilfredsstiller personlige preferanser), kan det være nødvendig å oppdatere bibliotekets filnavn i `save_usrlib` eller `online_usrlib`. **Unnlatelse 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.6.0.30
             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-2.40.so
               libsframe.so.0.0.0
               libhistory.so.8.2
               libncursesw.so.6.4
               libm.so.6
               libreadline.so.8.2
               libz.so.1.2.13
               $(cd /usr/lib; find libnss*.so* -type f)"

for BIN in $online_usrbin; do
    cp /usr/bin/$BIN /tmp/$BIN
    strip --strip-unneeded /tmp/$BIN
    install -vm755 /tmp/$BIN /usr/bin
    rm /tmp/$BIN
done

for LIB in $online_usrlib; do
    cp /usr/lib/$LIB /tmp/$LIB
    strip --strip-unneeded /tmp/$LIB
    install -vm755 /tmp/$LIB /usr/lib
    rm /tmp/$LIB
done

for i in $(find /usr/lib -type f -name \*.so* ! -name \*dbg) \
         $(find /usr/lib -type f -name \*.a)                 \
         $(find /usr/{bin,sbin,libexec} -type f); do
    case "$online_usrbin $online_usrlib $save_usrlib" in
        *$(basename $i)* )
            ;;
        * ) strip --strip-unneeded $i
            ;;
    esac
done

unset BIN LIB save_usrlib online_usrbin online_usrlib
```

Et stort antall filer vil bli rapportert som at filformatet ikke er gjenkjent. Disse advarslene kan trygt ignoreres. De indikerer at disse filene er skript i stedet for binære filer.

8.79. Rydde opp

Til slutt, 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 nødvendig lenger. Fjern den med:

```
find /usr -depth -name $(uname -m)-lfs-linux-gnu\* | xargs rm -rf
```

Til slutt fjerner du den midlertidige "tester" brukerkontoen som ble opprettet på begynnelsen av forrige kapittel.

```
userdel -r tester
```

Chapter 9. Systemkonfigurasjon

9.1. Introduksjon

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-20230101

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: mindre enn 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 oppført i `/etc/sysconfig/modules`, ved å bruke argumenter som også er gitt der

mountfs Monterer alle filsystemer, bortsett fra de som er merket *noauto*, eller er nettverksbasert

mountvirtfs Monterer virtuelle kjernefilsystemer, som f.eks `proc`

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 bootscriptene én etter én, 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 kjerneklokken til lokal tid i tilfelle maskinvareklokken 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 `/etc/sysctl.conf`, 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 `/dev` mappen og starter Udev nissen

udev_retry Forsøker på nytt mislykkede udev uevents, og kopier genererte regelfiler fra `/run/udev` til `/etc/udev/rules.d` hvis påkrevd

9.3. Oversikt over enhets- og modulhåndtering

I Kapittel 8, installerte vi udev nissen når eudev 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 maskinvareenhetene faktisk eksisterte. Dette ble vanligvis gjort via en **MAKEDEV** skript, som inneholder et antall anrop til **mknod** programmet med det aktuelle hoved- og mindre enhetsnumre for alle mulige enheter som kan eksistere i verden.

Ved å bruke udev metoden vil bare de enhetene som oppdages av kjernen få enhetsnoder opprettet for dem. Fordi disse enhetsnodene vil være opprettet hver gang systemet starter, vil de bli lagret på et `devtmpfs` filsystem (et virtuelt filsystem som ligger utelukkende i systemminnet). Enhetsnoder krever ikke mye plass, så minnet som brukes er ubetydelig.

9.3.1. Historie

I februar 2000 ble et nytt filsystem kalt `devfs` flettet inn i 2.3.46-kjernen og ble gjort tilgjengelig under 2.4-serien med stabile kjerner. Selv om den var til stede i selve kjernekilden, fikk denne metoden for å lage enheter dynamisk aldri overveldende støtte fra kjerneutviklere.

Hovedproblemet med tilnærmingen vedtatt av `devfs` var måten den håndterte enheten på ved oppdagelse, opprettelse og navngivning. Det siste problemet enhetsnavnet på en enhetsnode, var kanskje den mest kritiske. Det er generelt akseptert at hvis enhetsnavn kan konfigureres, så skal enhetsnavn politikken være opp til en systemadministrator, og ikke pålagt dem av en spesiell(e) utvikler(e). `devfs` filsystemet led også av kjøreforhold som var iboende i utformingen og ikke kunne fikses uten en betydelig revisjon av kjernen. `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 **udevd**. Basert på reglene spesifisert i filene i `/etc/udev/rules.d`, `/usr/lib/udev/rules.d`, og `/run/udev/rules.d` mappene, **udevd** vil opprette flere symbolkoblinger til enhetsnoden, eller endre tillatelsene, eieren eller gruppen, eller endre den interne **udevd** databaseoppføring (navn) for det objektet.

Reglene i disse tre mappene er nummererte og alle tre mappene slås sammen. Hvis **udevd** ikke finner en regel for enheten den oppretter, vil den opprettholde tillatelsene og eierskapet som `devtmpfs` brukte i utgangspunktet.

9.3.2.3. Modullasting

Enhetsdrivere kompilert som moduler kan ha innebygde aliaser. Aliaser er synlige i utdataene til **modinfo** programmet og er vanligvis relatert til de bussspesifikke identifikatorene til enheter støttet av en modul. For eksempel *snd-fm801* driveren støtter PCI-enheter med leverandør-ID 0x1319 og enhets-ID 0x0801, og har et alias "pci:v00001319d00000801sv*sd*bc04sc01i*". For de fleste enheter eksporterer bussdriveren aliaset til driveren som vil håndtere enheten via `sysfs`. F.eks, `/sys/bus/pci/devices/0000:00:0d.0/modalias` filen kan inneholde strengen "pci:v00001319d00000801sv00001319sd00001319bc04sc01i00". Standardreglene som følger med udev vil gjøre at **udevd** 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 lasting av uønskede drivere kan bli forhindret.

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 **udevd** som beskrevet ovenfor.

9.3.3. Problemer med å laste moduler og lage enheter

Det er noen mulige problemer når det kommer til å automatisk opprette enhetsnoder.

9.3.3.1. En kjernemodul lastes ikke automatisk

Udev vil bare laste en modul hvis den har et bussspesifikt alias og bussdriveren eksporterer de nødvendige aliasene til `sysfs`. I andre tilfeller bør man ordne modullasting på andre måter. Med Linux-6.1.11, 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-6.1.11, 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-maskinvaredrivere som f.eks *loop* i det hele tatt.

9.3.3.2. En kjernemodul lastes ikke automatisk, og udev er ikke beregnet på å laste den

Hvis 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 innpakkningen 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)` manualside 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 innpakningsmoduler, 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 Section 9.4, “Administrere enheter” og Section 9.5, “Generell nettverkskonfigurasjon” for eksempler.

9.3.4. Nyttig lesning

Ytterligere nyttig dokumentasjon er tilgjengelig på følgende nettsteder:

- En brukerromsimplementering av `devfs` *http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf*
- `sysfs` filsystemet *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 Section 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
```

Note

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 brukes.

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 nøkler, atskilt med komma og valgfritt mellomrom. Denne regelens nøkler og en forklaring av hver av dem er som følger:

- `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 ignorer VLAN- eller et bro undergrensesnitt (fordi disse undergrensesnittene ikke har drivere). Disse undergrensesnittene hoppes 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.

9.4.2. CD-ROM symbolkoblinger

Noe programvare som du kanskje vil installere senere (f.eks. diverse mediespillere) 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 identifikasjonsstrengene 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.

Important

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 Section 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 Section 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+="tvtuner"

EOF
```

Resultatet er at `/dev/video0` og `/dev/video1` enheter refererer fortsatt tilfeldig til tuneren og webkameraet (og bør derfor aldri brukes direkte), men det finnes symbolkoblinger `/dev/tvtuner` og `/dev/webcam` som alltid peker på den rette enheten.

9.5. 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*.

Note

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 grensesnittetnavnet 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 Section 8.70, “Eudev-3.2.11”) 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 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 subnettet. Hver oktett i en IP-adresse er 8 bits. Hvis subnettets nettmaske 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 én oppføring (krever sekundære servere for reservefunksjon). Hvis du bare trenger eller vil ha én DNS-server, fjern andre *nameserver* linjen fra filen. IP adressen kan også være en ruter på det lokale nettverket.

Note

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 "<lfs>" > /etc/hostname
```

`<lfs>` 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 IP adressen, fullt kvalifisert domenenavn (FQDN) og mulige aliaser for bruk i filen `/etc/hosts`. Syntaks er:

```
IP_addresse minvert.eksempel.org aliaser
```

Med mindre datamaskinen skal være synlig for Internett (dvs. det er et registrert domene og en gyldig blokk med tildelte IP adresser—de fleste brukere har ikke dette), sørg for at IP adressen er i den private nettverkets IP adresseområde. Gyldige områder er:

```
Privat nettverk Adresseområde      Normalt prefiks
10.0.0.1 - 10.255.255.254           8
172.x.0.1 - 172.x.255.254           16
192.168.y.1 - 192.168.y.254         24
```

x kan være et hvilket som helst tall i området 16-31. y kan være et hvilket som helst tall i område 0-255.

En gyldig privat IP adresse kan være 192.168.1.1. Et gyldig FQDN for denne IPen kan være lfs.example.org.

Selv om du ikke bruker et nettverkskort, kreves det fortsatt et gyldig FQDN. Dette er nødvendig for at enkelte programmer skal fungere korrekt.

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>
<192.168.1.1> <FQDN> <HOSTNAME> [alias1] [alias2 ...]
::1       localhost ip6-localhost ip6-loopback
ff02::1   ip6-allnodes
ff02::2   ip6-allrouters

# End /etc/hosts
EOF
```

<192.168.1.1>, *<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 diskuter 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), og 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 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
```

Note

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 "reservert".

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:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S06:once:/sbin/sulogin
s1:1:respawn:/sbin/sulogin

1:2345:respawn:/sbin/agetty --noclear tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab
EOF
```

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 spørsmålstegnet 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 **udevd**, 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 **udevd** 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/rtc.

For informasjon om lasting av kjernemoduler og udev, se Section 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.

Note

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 skjermskrift 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)` og `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 **dmesg -n**. Gyldige nivåer er fra "1" (ingen meldinger) til "8". Standardnivået er "7".

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 nøkkelkoder 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 konverter 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:

- 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 tastaturoppsett). Linux-6.1.11 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 “alpha”. 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).

Note

`/etc/sysconfig/console` filen kontroller kun lokaliseringen av Linux-tekstkonsollen. Det har ingenting med innstilling å gjøre riktig tastaturoppsett 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` konfigurasjonsskriptet. 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 tidsstempelet **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 sysklogd parameters
#SYSKLOGD_PARMS="-m 0"

# Console parameters
```

```
#UNICODE=1
#KEYMAP="de-latin1"
#KEYMAP_CORRECTIONS="euro2"
#FONT="lat0-16 -m 8859-15"
#LEGACY_CHARSET=
```

9.6.8.1. 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 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. Oppstartsfilene til Bash skallet

Skallprogrammet **/bin/bash** (heretter referert til som “skallet”) bruker en samling oppstartsfiler for å hjelpe til å lage et miljø å kjøre i. Hver fil har en spesifikk bruk og kan påvirke pålogging og interaktive miljøer annerledes. Filene i `/etc` mappen gir globale innstillinger. Hvis en tilsvarende fil finnes i hjemmemappen, kan den 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 startet på kommandolinjen (f.eks., `[prompt]$`**/bin/bash**). Et ikke-interaktiv skall er vanligvis tilstede når et skallskript kjører. Det er ikke-interaktivt fordi det behandler et skript og ikke venter på brukerinndata mellom kommandoer.

For mer informasjon, se *Bash Startup Files* og *Interactive Shells* seksjonene i *Bash Features* kapittel på Bash infosidene (**info bash**).

Filene `/etc/profile` og `~/.bash_profile` leses når skallet er startet som et interaktivt påloggingsskall.

Grunnfilen `/etc/profile` nedenfor setter noen miljøvariabler som er nødvendige for morsmålsstøtte. Å sette de riktig resulterer i:

- Utdataene fra programmer oversatt til morsmålet
- Riktig klassifisering av tegn i bokstaver, sifre og andre klasser. Dette er nødvendig for at **bash** skal akseptere ordentlig ikke-ASCII tegn i kommandolinjer i ikke-engelske språk
- Riktig alfabetisk sorteringsrekkefølge for landet
- Passende standard papirstørrelse
- 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 landet (f.eks., “GB”). *<charmap>* bør erstattes med den kanoniske tegntabellen for din valgte lokalitet. Valgfri modifikatorer som f.eks “@euro” kan også være tilstede.

Listen over alle lokaliteter som støttes av Glibc kan fås ved å kjøre følgende kommando:

```
locale -a
```

Tegntabeller 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. kreves det at “UTF-8” er skrevet som “UTF-8”, ikke “utf8”), så det er tryggest i de fleste tilfeller for 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” lokaliteten, kommandoen over vil skrive ut:

```
ISO-8859-1
```

Dette resulterer er en endelig lokaleinnstilling av “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, tegnkoding som brukes av lokaliteten, den lokale valutaen og prefikset før telefonnummeret for å ringe 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 var installert i Section 8.5, “Glibc-2.37” eller 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** kommandoen, 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, å undersøke hvordan andre Linux distribusjoner støtter lokaliteten din kan gi noe nyttig informasjon.

Når de riktige lokale innstillingene er bestemt, oppretter du `/etc/profile` filen:

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

export LANG=<ll>_<CC>.<charmap><@modifiers>

# End /etc/profile
EOF
```

“C” (standard) og “en_US.utf8” (det anbefalte for engelske brukere i USA) lokalitetene er forskjellige. “C” bruker US-ASCII 7-biters tegnsett, og behandler byte med det høye bitsettet “på” som ugyldige tegn. Det er derfor, f.eks **ls** kommandoen erstatter dem med spørsmålstegn i det lokalet. Også et forsøk på å sende post med slike tegn fra Mutt eller Pine resulterer i ikke-RFC samsvars meldinger som sendes (tegnsettet i den utgående posten er indikert som “unknown 8-bit”). Så du kan bruke “C” lokaliteten bare hvis du er sikker på at du aldri trenger 8-bits tegn.

UTF-8 baserte lokaliteter støttes ikke godt av noen programmer. Det pågår arbeid med å dokumentere og om mulig fikse slike problemer, se *https://www.linuxfromscratch.org/blfs/view/11.3/introduction/locale-issues.html*.

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
```

Chapter 10. Gjøre LFS systemet oppstartbart

10.1. Introduksjon

Det er på tide å gjøre LFS systemet oppstartbart. Dette kapittelet diskuterer å opprette `/etc/fstab` filen, bygge en kjerne for det nye LFS systemet, og installere GRUB oppstartslasteren slik at LFS systemet kan velges for oppstart ved oppstart.

10.2. Opprette /etc/fstab filen

`/etc/fstab` filen brukes av noen programmer til bestemme hvor filsystemer skal monteres som standard, i hvilken rekkefølge, og hvilke som må kontrolleres (for integritetsfeil) før montering. Lag en ny filsystemtabell som denne:

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab

# file system  mount-point  type     options             dump  fsck
#                                                              order

/dev/<xxx>     /            <fff>    defaults            1     1
/dev/<yyy>     swap         swap     pri=1               0     0
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

# End /etc/fstab
EOF
```

Erstatt *<xxx>*, *<yyy>*, og *<fff>* med verdiene som passer for systemet, for eksempel, `sda2`, `sda5`, og `ext4`. For detaljer om de seks feltene i denne filen, se **man 5 fstab**.

Filsystemer med MS-DOS eller Windows opprinnelse (dvs. vfat, ntfs, smbfs, cifs, iso9660, udf) trenger et spesielt alternativ, utf8, for ikke-ASCII tegn i filnavn som skal tolkes riktig. For ikke-UTF-8-lokaliteter, verdien av `iocharset` bør settes til å være det samme som tegnsettet for lokaliteten, justert på en slik måte at kjernen forstår det. Dette fungerer hvis den relevante tegnsettdefinisjonen (funnet under 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-6.1.11

Linux pakken inneholder Linux kjernen.

Omtrentlig byggetid: 1.5 - 130.0 SBU (typically omtrent 12 SBU)
Nødvendig diskplass: 1200 - 8800 MB (typically omtrent 1700 MB)

10.3.1. Installasjon av kjernen

Å bygge kjernen innebærer noen få trinn—konfigurasjon, kompilering og installasjon. Les `README` filen i kjernekildetreet for alternative metoder til måten denne boken konfigurerer kjernen på.

Forbered 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 menyconfig ncurses grensesnitt linjetegning på en UTF-8 linux tekstkonsoll.

Hvis brukt, sørg for å erstatte *`<host_LANG_value>`* med verdien av `$LANG` variabel fra verten din. Du kan alternativt bruke vertens verdi av `$LC_ALL` eller `$LC_CTYPE`.

make menuconfig

Dette starter et ncurses menydrevet grensesnitt. For andre (grafiske) grensesnitt, skriv **make help**.

For generell informasjon om kjernekonfigurasjon se *https://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt*. BLFS har litt informasjon angående spesielle kjernekonfigurasjonskrav for pakker utenfor LFS på *https://www.linuxfromscratch.org/blfs/view/11.3/longindex.html#kernel-config-index*. Ytterligere informasjon om konfigurering og bygging av kjernen finner du på *http://www.kroah.com/lkn/*

Note

Et godt utgangspunkt for å sette opp kjernekonfigurasjonen er å kjøre **make defconfig**. Dette vil sette grunnleggende konfigurasjon til en god tilstand som tar din nåværende systemarkitektur i betraktning.

Sørg for å aktivere/deaktivere/stille inn følgende funksjoner, ellers kan systemet ikke fungere riktig eller starte opp i det hele tatt:

```
Processor type and features --->
   [*] Build a relocatable kernel [CONFIG_RELOCATABLE]
   [*]   Randomize the address of the kernel image (KASLR) [CONFIG_RANDOMIZE_BASE]
General setup --->
   [ ] Compile the kernel with warnings as errors [CONFIG_WERROR]
   < > Enable kernel headers through /sys/kernel/kheaders.tar.xz [CONFIG_IKHEADERS]
General architecture-dependent options  --->
   [*] Stack Protector buffer overflow detection [CONFIG_STACKPROTECTOR]
   [*]   Strong Stack Protector [CONFIG_STACKPROTECTOR_STRONG]
Device Drivers  --->
  Graphics support --->
   Frame buffer Devices --->
      <*> Support for frame buffer devices --->
   Console display driver support --->
      [*] Framebuffer Console support [CONFIG_FRAMEBUFFER_CONSOLE]
  Generic Driver Options  --->
   [ ] Support for uevent helper [CONFIG_UEVENT_HELPER]
   [*] Maintain a devtmpfs filesystem to mount at /dev [CONFIG_DEVTMPFS]
   [*]   Automount devtmpfs at /dev, after the kernel mounted the rootfs [CONFIG_DEVTMPFS_MOUNT]
```

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 [CONFIG_X86_X2APIC]
Device Drivers --->
  [*] PCI Support ---> [CONFIG_PCI]
    [*] Message Signaled Interrupts (MSI and MSI-X) [CONFIG_PCI_MSI]
  [*] IOMMU Hardware Support ---> [CONFIG_IOMMU_SUPPORT]
    [*] Support for Interrupt Remapping [CONFIG_IRQ_REMAP]
```

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* (https://www.linuxfromscratch.org/blfs/view/11.3/longindex.html#kernel-config-index).

Note

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*.

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.

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/Eudev.

Maintain a devtmpfs

Dette vil opprette automatiserte enhetsnoder som er befolket av kjerne, selv uten at Udev kjører. Udev kjører så på toppen av dette, administrere tillatelser og legge til symbolkoblinger. Denne konfigurasjonen element er nødvendig for alle brukere av Udev/Eudev.

Automount devtmpfs at /dev

Dette vil montere kjernevisningen til enhetene på /dev ved bytte til rotfilsystem rett før start av init.

Framebuffer Console support

Dette er nødvendig for å vise Linux konsollen på en ramme bufferenhet. For å la kjernen skrive ut feilsøkingsmeldinger på et tidlig oppstartsstadium, bør den ikke bygges som en kjernemodul med mindre en initramfs vil bli brukt. Og hvis `CONFIG_DRM` (Direct Rendering Manager) er aktivert, er det sannsynlig at `CONFIG_DRM_FBDEV_EMULATION` (Enable legacy fbdev support for your modesetting driver) bør være aktivert også.

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-6.1.11` mappen. Derimot, vi anbefaler ikke dette alternativet. Det er ofte bedre å utforske alle konfigurasjonsmenyer og lage kjernekonfigurasjonen fra grunnen av.

Kompiler kjernebildet og modulene:

```
make
```

Hvis du bruker kjernemoduler, modulkonfigurasjon i `/etc/modprobe.d` kan være nødvendig. Informasjon knyttet til moduler og kjernekonfigurasjon er lokalisert i Section 9.3, “Oversikt over enhets- og modulhåndtering” og kjerne dokumentasjon i `linux-6.1.11/Documentation` mappen. Også, `modprobe.d(5)` kan være av interesse.

Med mindre modulstøtte er deaktivert i kjernekonfigurasjonen, installere modulene med:

```
make modules_install
```

Etter at kjernekompileringen er fullført, er flere trinn nødvendig for å fullføre installasjonen. Noen filer må kopieres til `/boot` mappen.

Caution

Hvis 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 gjør det er å opprette oppføringen 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-6.1.11-lfs-11.3
```

`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-6.1.11
```

Kjernens konfigurasjonsfil `.config` produsert av **make menuconfig** steget ovenfor inneholder alle konfigurasjonsvalgene for kjernen som nettopp ble kompilert. Det er en god idé å beholde denne filen for fremtidig referanse:

```
cp -iv .config /boot/config-6.1.11
```

Installer dokumentasjonen for Linux kjernen:

```
install -d /usr/share/doc/linux-6.1.11
cp -r Documentation/* /usr/share/doc/linux-6.1.11
```

Det er viktig å merke seg at filene i kjernekildens mappen ikke eies av *root*. Når en pakke pakkes ut som bruker *root* (som vi gjorde inne i chroot), filene har bruker- og gruppe-IDer for hva som helst de var på pakkerens datamaskin. Dette er vanligvis ikke et problem for enhver pakke som skal installeres fordi kildetreet blir fjernet etter installasjonen. Imidlertid er Linux kildetreet ofte beholdt i lang tid. På grunn av dette er det en sjanse at hvilken bruker-ID pakkeren brukte vil bli tildelt noen på maskinen. Den personen ville da ha skrivetilgang til kjernens kilde.

Note

I mange tilfeller må konfigurasjonen av kjernen være oppdatert for pakker som vil bli installert senere i BLFS. I motsetning til andre pakker, er det ikke nødvendig å fjerne kjernekildetreet etter at den nybygde kjernen er installert.

Hvis kjernekildetreet skal beholdes, kjør **chown -R 0:0** på `linux-6.1.11` mappen å sikre at alle filer eies av brukeren *root*.

Warning

Noe kjernedokumentasjon anbefaler å opprette en symbolkobling fra `/usr/src/linux` som peker på kjernens kildemappe. Dette er spesifikt for kjerner før 2.6-serien og *må ikke* opprettes på et LFS system, for det kan forårsake problemer for pakker du kanskje ønsker å bygge når basis LFS systemet er fullstendig.

Warning

Deklarasjonene i systemets `include` mappe (`/usr/include`) bør *alltid* være de som Glibc ble kompilert mot, det vil si de desinfiserte deklarasjonene installert i Section 5.4, “Linux-6.1.11 API Deklarasjoner”. Derfor bør de *aldri* erstattes av enten de rå kjernedeklarasjonene eller andre kjernerensede deklarasjoner.

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-6.1.11, vmlinuz-6.1.11-lfs-11.3, og System.map-6.1.11
Installerte mapper: /lib/modules, /usr/share/doc/linux-6.1.11

Korte beskrivelser

`config-6.1.11` — Inneholder alle konfigurasjonsvalgene for kjernen

`vmlinuz-6.1.11-lfs-11.3` — Motoren til Linux systemet. Når du slår på datamaskinen, er kjernen den første delen av operativsystemet som blir lastet. Den oppdager og initialiserer alle komponenter i datamaskinens maskinvare, gjør deretter disse komponentene tilgjengelige som et tre med filer til programvarer og forvandler en enkelt CPU til en multitasking-maskin med å kjøre mange programmer tilsynelatende samtidig

`System.map-6.1.11` — En liste over adresser og symboler; den kartlegger inngangspunktene og adresser til alle funksjonene og datastrukturene i kjernen

10.4. Bruke GRUB til å sette opp oppstartsprosessen

Note

Hvis systemet ditt har UEFI støtte og du ønsker å starte LFS med UEFI, bør du hoppe over denne siden og konfigurere GRUB med UEFI støtte ved å bruke instruksjonene i *BLFS siden*.

10.4.1. Introduksjon

Warning

Å konfigurere GRUB feil kan gjøre systemet ditt ubrukelig uten en alternativ oppstartsenhet som en CD-ROM eller oppstartbar USB-stasjon. Denne delen er ikke nødvendig for å starte opp LFS systemet. Du kan bare endre din nåværende oppstartslaster, f.eks. Grub-Legacy, GRUB2 eller LILO.

Sørg for at en nødoppstartsdiskett er klar til å "redde" datamaskinen hvis datamaskinen blir ubrukelig (ikke-oppstartbar). Hvis du ikke allerede har en oppstartsenhet, kan du opprette en. For at prosedyren nedenfor skal fungere, så må du hoppe til BLFS og installere **`xorriso`** fra *libisoburn* pakken.

```
cd /tmp
grub-mkrescue --output=grub-img.iso
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

10.4.2. GRUB navnekonvensjoner

GRUB bruker sin egen navnestruktur for stasjoner og partisjoner i formen *(hdn,m)*, hvor *n* er harddisknummeret og *m* er partisjonensnummeret. Harddisknummeret starter fra null, men partisjonsnummeret starter fra 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 hardisk. Dette området er ikke en del av noe filsystem. programmene der gir tilgang til GRUB moduler i oppstartspartisjonen. Standardplasseringen er /boot/grub/.

Plasseringen av oppstartspartisjonen er et valg av brukeren som påvirker konfigurasjonen. En anbefaling er å ha en egen liten (foreslått størrelse er 200 MB) partisjon kun for oppstartsinformasjon. På den måten 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:

Warning

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
```

Note

Hvis systemet har blitt startet opp med UEFI, **grub-install** vil prøve å installere filer for *x86_64-efi* målet, men disse filene er ikke installert i 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 ext2
set root=(hd0,2)

menuentry "GNU/Linux, Linux 6.1.11-lfs-11.3" {
        linux   /boot/vmlinuz-6.1.11-lfs-11.3 root=/dev/sda2 ro
}
EOF
```

Note

Fra GRUB sitt perspektiv, kjernefilene er i forhold til partisjonen som brukes. Hvis du brukte en separat /boot-partisjon, fjern /boot fra ovenstående *linux* linjen. Du må også endre *set root* linjen for å peke på oppstartspartisjonen.

Note

GRUB betegnelsen for en partisjon kan endres hvis du la til eller fjernet noen disker (inkludert flyttbare disker 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 enhetsnoden 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.

Caution

Det er en kommando, grub-mkconfig, som kan skrive en konfigurasjonsfil automatisk. Den bruker et sett med skript i /etc/grub.d/ og vil ødelegge eventuelle tilpasninger du gjør. Disse skriptene er først og fremst designet for distribusjoner uten kilder og anbefales ikke for LFS. Hvis du installerer en kommersiell Linux distribusjon, er det en god sjanse at dette programmet skal kjøres. Sørg for å sikkerhetskopiere grub.cfg filen.

Chapter 11. Slutten

11.1. Slutten

Bra gjort! Det nye LFS systemet er installert! Vi ønsker deg mye suksess med ditt skinnende nye spesialbygde Linux-system.

Det kan være lurt å 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 11.3 > /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="11.3"
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="11.3"
ID=lfs
PRETTY_NAME="Linux From Scratch 11.3"
VERSION_CODENAME="<your name here>"
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.
- 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 $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 11.3* automatisk.

Når omstarten er fullført, er LFS systemet klart til bruk og mer programvare kan legges til for å passe dine behov.

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 Chapter 1 - Mailing Lists for mer informasjon.

- Linux dokumentasjonsprosjektet

  Målet med Linux dokumentasjonsprosjektet (TLDP) er å samarbeide om alle problemene med Linux dokumentasjon. TLDP funksjonene en stor samling av HOWTOer, guider og 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 alle miljøer. 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
  mkdir -pv $LFS/$(readlink $LFS/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. Vår 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=/bin:/usr/bin:/sbin:/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* og *wget* 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.

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**+$\leftarrow$ og **Alt**+$\rightarrow$ 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.

Note

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**.

Part V. Vedlegg

Appendix A. Akronymer og begreper

ABI Binært applikasjonsgrensesnitt (Application Binary Interface)

ALFS Automatisert Linux fra bunnen (Automated Linux From Scratch)

API Applikasjonsprogrammeringsgrensesnitt (Application Programming Interface)

ASCII Amerikansk standardkode for informasjonsutveksling (American Standard Code for Information Interchange)

BIOS Grunnleggende system for inndata/utdata (Basic Input/Output System)

BLFS Utover Linux fra bunnen (Beyond Linux From Scratch)

BSD Berkeley programvaredistribusjon (Berkeley Software Distribution)

chroot endre rot (change root)

CMOS Komplementær metalloksyd halvleder (Complementary Metal Oxide Semiconductor)

COS Tjenesteklasse (Class Of Service)

CPU Sentral prosesseringsenhet (Central Processing Unit)

CRC Syklisk redundanssjekk (Cyclic Redundancy Check)

CVS System for samtidige versjoner (Concurrent Versions System)

DHCP Dynamisk vertskonfigurasjonsprotokoll (Dynamic Host Configuration Protocol)

DNS Domenenavntjeneste (Domain Name Service)

EGA Forbedret grafikkadapter (Enhanced Graphics Adapter)

ELF Kjørbart og koblingsbart format (Executable and Linkable Format)

EOF Slutt på fil (End of File)

EQN ligning (equation)

ext2 andre utvidede filsystemet (second extended file system)

ext3 tredje utvidede filsystemet (third extended file system)

ext4 fjerde utvidede filsystemet (fourth extended file system)

FAQ Ofte stilte spørsmål (Frequently Asked Questions)

FHS Standard for Filsystemhierarkiet (Filesystem Hierarchy Standard)

FIFO Først inn først ut (First-In, First Out)

FQDN Fullt kvalifisert domenenavn (Fully Qualified Domain Name)

FTP Filoverføringsprotokoll (File Transfer Protocol)

GB Gigabyte (Gigabytes)

GCC GNU kompilatorsamling (GNU Compiler Collection)

GID Gruppeidentifikator (Group Identifier)

GMT Greenwich gjennomsnittstid (Greenwich Mean Time)

HTML Hypertekst markeringsspråk (Hypertext Markup Language)

IDE Integrert drivelektronikk (Integrated Drive Electronics)

IEEE Institutt for elektriske og elektroniske ingeniører (Institute of Electrical and Electronic Engineers)
IO Inndata/utdata (Input/Output)
IP Internett protokoll (Internet Protocol)
IPC Kommunikasjon mellom prosesser (Inter-Process Communication)
IRC Internett relé nettprat (Internet Relay Chat)
ISO Internasjonal organisasjon for standardisasjon (International Organization for Standardization)
ISP Internett tjenesteleverandør (Internet Service Provider)
KB Kilobyte (Kilobytes)
LED Lysemitterende diode (Light Emitting Diode)
LFS Linux fra bunnen (Linux From Scratch)
LSB Linux standardbase (Linux Standard Base)
MB Megabyte (Megabytes)
MBR Master oppstart opptak (Master Boot Record)
MD5 Meldingssammendrag 5 (Message Digest 5)
NIC Nettverksgrensesnittkort (Network Interface Card)
NLS Støtte for morsmål (Native Language Support)
NNTP Transportprotokoll for nettverksnyheter (Network News Transport Protocol)
NPTL Innebygd POSIX trådbibliotek (Native POSIX Threading Library)
OSS Åpent lydsystem (Open Sound System)
PCH Forhåndskompilerte deklarasjonsfiler (Pre-Compiled Headers)
PCRE Perlkompatibelt regulært uttrykk (Perl Compatible Regular Expression)
PID Prosessidentifikator (Process Identifier)
PTY pseudoterminal (pseudo terminal)
QOS Tjenestekvalitet (Quality Of Service)
RAM Tilfeldig tilgangsminne (Random Access Memory)
RPC Anrop for ekstern prosedyre (Remote Procedure Call)
RTC Sanntidsklokke (Real Time Clock)
SBU Standard byggeenhet (Standard Build Unit)
SCO Santa Cruz operasjonen (The Santa Cruz Operation)
SHA1 Sikker nøkkel algoritme 1 (Secure-Hash Algorithm 1)
TLDP Linux dokumentasjonsprosjekt (The Linux Documentation Project)
TFTP Triviell filoverføringsprotokoll (Trivial File Transfer Protocol)
TLS Trådlokal lagring (Thread-Local Storage)
UID Brukeridentifikator (User Identifier)
umask maske for opprettelse av brukerfil (user file-creation mask)
USB Universell seriebuss (Universal Serial Bus)

UTC Koordinert universell tid (Coordinated Universal Time)

UUID Universell unik identifikator (Universally Unique Identifier)

VC Virtuell konsoll (Virtual Console)

VGA Videografikkmatrise (Video Graphics Array)

VT Virtuell terminal (Virtual Terminal)

Appendix B. Anerkjennelser

Vi vil takke følgende personer og organisasjoner for deres bidrag til Linux From Scratch Project.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – LFS skaper
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – LFS Administrerende Redaktør
- *Jim Gifford* <jim@linuxfromscratch.org> – CLFS Prosjekt Medleder
- *Pierre Labastie* <pierre@linuxfromscratch.org> – BLFS redaktør og ALFS leder
- *DJ Lucas* <dj@linuxfromscratch.org> – LFS og BLFS redaktør
- *Ken Moffat* <ken@linuxfromscratch.org> – BLFS redaktør
- Utallige andre personer på de ulike LFS og BLFS postlistene som bidro til å gjøre denne boken mulig ved å gi sine forslag, teste boken, og sende inn feilrapporter, instruksjoner og deres erfaringer med installasjon av ulike pakker.

Oversettere

- *Manuel Canales Esparcia* <macana@macana-es.com> – Spansk LFS oversettelsesprosjekt
- *Johan Lenglet* <johan@linuxfromscratch.org> – Fransk LFS oversettelsesprosjekt 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> – Portuguese LFS oversettelsesprosjekt
- *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 Phermsawang* <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 McMurchy* <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

Appendix C. Avhengigheter

Hver pakke bygget i LFS er avhengig av en eller flere andre pakker for å bygges og installeres riktig. Noen pakker deltar til og med i sirkulære avhengigheter, det vil si at den første pakken avhenger av den andre i sin tur avhenger av den første. På grunn av disse avhengighetene er rekkefølgen som pakkene bygges i LFS veldig viktig. Formålet med denne siden er å dokumentere avhengighetene til hver pakke bygget i LFS.

For hver pakke som bygges er det tre, og noen ganger opptil fem typer avhengigheter oppført nedenfor. Den første viser hvilke andre pakker må være tilgjengelig for å kompilere og installere den aktuelle pakken. Den andre viser pakkene som må være tilgjengelige når noen programmer eller biblioteker fra pakken brukes under kjøring. Den tredje viser hvilke pakker, i tillegg til de på den første listen, 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. I de fleste tilfeller er dette fordi disse pakkene hardkoder kodebaner til binærfiler i skriptene deres. Hvis ikke dette blir bygget i en bestemt rekkefølge, kan det føre til at stier til /tools/bin/[binær] blir plassert inne i skript installert i det endelige systemet. Dette er åpenbart ikke ønskelig.

Den siste listen over avhengigheter er valgfrie pakker som ikke er adressert i LFS, men kan være nyttig for brukeren. Disse pakkene kan ha ekstra obligatoriske eller valgfrie avhengigheter. For disse avhengigheter, er den anbefalte praksisen å installere dem etter fullføring av LFS boken og gå tilbake og gjenoppbygg LFS pakken. I flere tilfeller, er reinstallasjon adressert i BLFS.

Acl

Installasjonen avhenger av: Attr, Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed, og Texinfo

Påkrevd ved kjøretid: Attr og Glibc

Testpakke avhenger av: Automake, Diffutils, Findutils, og Libtool

Må installeres før: Coreutils, Sed, Tar, og Vim

Valgfrie avhengigheter: Ingen

Attr

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Automake, Diffutils, Findutils, og Libtool

Må installeres før: Acl og Libcap

Valgfrie avhengigheter: Ingen

Autoconf

Installasjonen avhenger av: Bash, Coreutils, Grep, M4, Make, Perl, Sed, og Texinfo

Påkrevd ved kjøretid: Bash, Coreutils, Grep, M4, Make, Sed, og Texinfo

Testpakke avhenger av: Automake, Diffutils, Findutils, GCC, og Libtool

Må installeres før: Automake

Valgfrie avhengigheter: *Emacs*

Automake

Installasjonen avhenger av: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed, og Texinfo

Påkrevd ved kjøretid: Bash, Coreutils, Grep, M4, Sed, og Texinfo

Testpakke avhenger av: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, og Tar

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Bash

Installasjonen avhenger av: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc, Ncurses, og Readline

Testpakke avhenger av: Expect og Shadow

Må installeres før: Ingen

Valgfrie avhengigheter: *Xorg*

Bc

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Grep, 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, Sed, Texinfo, og Zlib

Påkrevd ved kjøretid: Glibc og Zlib

Testpakke avhenger av: DejaGNU og Expect

Må installeres før: Ingen

Valgfrie avhengigheter: *Elfutils* og *Jansson*

Bison

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Diffutils, Findutils, og Flex

Må installeres før: Kbd og Tar

Valgfrie avhengigheter: *Doxygen*

Bzip2

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, og Patch

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen

Må installeres før: File

Valgfrie avhengigheter: Ingen

Check

Installasjonen avhenger av: Gawk, GCC, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Bash og Gawk

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Coreutils

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Libcap, Make, OpenSSL, Patch, Perl, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Diffutils, E2fsprogs, Findutils, Shadow, og Util-linux

Må installeres før: Bash, Diffutils, Eudev, Findutils, og Man-DB

Valgfrie avhengigheter: *Expect.pm* og *IO::Tty*

DejaGNU

Installasjonen avhenger av: Bash, Coreutils, Diffutils, Expect, GCC, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Expect og Bash

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Diffutils

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Perl

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

E2fsprogs

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Sed, Texinfo, og Util-linux

Påkrevd ved kjøretid: Glibc og Util-linux

Testpakke avhenger av: Procps-ng og Psmisc

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Eudev

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Gperf, Make, Sed, og Util-linux

Påkrevd ved kjøretid: Glibc, Kmod, Xz, Util-linux, og Zlib.

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Expat

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen

Må installeres før: Python og XML::Parser

Valgfrie avhengigheter: Ingen

Expect

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, og Tcl

Påkrevd ved kjøretid: Glibc og Tcl

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: *Tk*

File

Installasjonen avhenger av: Bash, Binutils, Bzip2, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, Xz, og Zlib

Påkrevd ved kjøretid: Glibc, Bzip2, Xz, og Zlib

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: *libseccomp*

Findutils

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Bash og Glibc

Testpakke avhenger av: DejaGNU, Diffutils, og Expect

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Flex

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed, og Texinfo

Påkrevd ved kjøretid: Bash, Glibc, og M4

Testpakke avhenger av: Bison og Gawk

Må installeres før: Binutils, IProute2, Kbd, Kmod, og Man-DB

Valgfrie avhengigheter: Ingen

Gawk

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, MPFR, Patch, Readline, Sed, og Texinfo

Påkrevd ved kjøretid: Bash, Glibc, og Mpfr

Testpakke avhenger av: Diffutils

Må installeres før: Ingen

Valgfrie avhengigheter: *libsigsegv*

GCC

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, Texinfo, og Zstd

Påkrevd ved kjøretid: Bash, Binutils, Glibc, Mpc, og Python

Testpakke avhenger av: DejaGNU, Expect, og Shadow

Må installeres før: Ingen

Valgfrie avhengigheter: *GNAT* og *ISL*

GDBM

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make, og Sed

Påkrevd ved kjøretid: Bash, Glibc, og Readline

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Gettext

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Sed, og Texinfo

Påkrevd ved kjøretid: Acl, Bash, Gcc, og Glibc

Testpakke avhenger av: Diffutils, Perl, og Tcl

Må installeres før: Automake og Bison

Valgfrie avhengigheter: Ingen

Glibc

Installasjonen avhenger av: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Linux API Headers, Make, Perl, Python, Sed, og Texinfo

Påkrevd ved kjøretid: Ingen

Testpakke avhenger av: File

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

GMP

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed, og Texinfo

Påkrevd ved kjøretid: GCC og Glibc

Testpakke avhenger av: Ingen

Må installeres før: MPFR og GCC

Valgfrie avhengigheter: Ingen

Gperf

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, og Make

Påkrevd ved kjøretid: GCC og Glibc

Testpakke avhenger av: Diffutils og Expect

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Grep

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Gawk

Må installeres før: Man-DB

Valgfrie avhengigheter: *PCRE2* og *libsigsegv*

Groff

Installasjonen avhenger av: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed, og Texinfo

Påkrevd ved kjøretid: GCC, Glibc, og Perl

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Man-DB og Perl

Valgfrie avhengigheter: *ghostscript* og *Uchardet*

GRUB

Installasjonen avhenger av: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo, og Xz

Påkrevd ved kjøretid: Bash, GCC, Gettext, Glibc, Xz, og Sed.

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Gzip

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Bash og Glibc

Testpakke avhenger av: Diffutils og Less

Må installeres før: Man-DB

Valgfrie avhengigheter: Ingen

Iana-Etc

Installasjonen avhenger av: Coreutils

Påkrevd ved kjøretid: Ingen

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Perl

Valgfrie avhengigheter: Ingen

Inetutils

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo, og Zlib

Påkrevd ved kjøretid: GCC, Glibc, Ncurses, og Readline

Testpakke avhenger av: Ingen

Må installeres før: Tar

Valgfrie avhengigheter: Ingen

Intltool

Installasjonen avhenger av: Bash, Gawk, Glibc, Make, Perl, Sed, og XML::Parser

Påkrevd ved kjøretid: Autoconf, Automake, Bash, Glibc, Grep, Perl, og Sed

Testpakke avhenger av: Perl

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

IProute2

Installasjonen avhenger av: Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, Libcap, Libelf, Linux API Headers, og Zlib

Påkrevd ved kjøretid: Bash, Coreutils, Glibc, Libcap, Libelf, og Zlib

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: *Berkeley DB* og *iptables*

Kbd

Installasjonen avhenger av: Bash, Binutils, Bison, Check, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch, og Sed

Påkrevd ved kjøretid: Bash, Coreutils, og Glibc

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Kmod

Installasjonen avhenger av: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, OpenSSL, Pkg-config, Sed, Xz, og Zlib

Påkrevd ved kjøretid: Glibc, Xz, og Zlib

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Eudev

Valgfrie avhengigheter: Ingen

Less

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, og Sed

Påkrevd ved kjøretid: Glibc og Ncurses

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Gzip

Valgfrie avhengigheter: *PCRE2* eller *PCRE*

Libcap

Installasjonen avhenger av: Attr, Bash, Binutils, Coreutils, GCC, Glibc, Perl, Make, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen

Må installeres før: IProute2 og Shadow

Valgfrie avhengigheter: *Linux-PAM*

Libelf

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, og Make

Påkrevd ved kjøretid: Glibc og Zlib

Testpakke avhenger av: Ingen

Må installeres før: IProute2 og Linux

Valgfrie avhengigheter: Ingen

Libffi

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Make, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: DejaGnu

Må installeres før: Python

Valgfrie avhengigheter: Ingen

Libpipeline

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Check

Må installeres før: Man-DB

Valgfrie avhengigheter: Ingen

Libtool

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Autoconf, Automake, Bash, Binutils, Coreutils, File, GCC, Glibc, Grep, Make, og Sed

Testpakke avhenger av: Autoconf, Automake, og Findutils

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Linux

Installasjonen avhenger av: Bash, Bc, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Libelf, Make, Ncurses, OpenSSL, Perl, og Sed

Påkrevd ved kjøretid: Ingen

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: *cpio* og *LLVM* (with Clang)

Linux API Headers

Installasjonen avhenger av: Bash, Binutils, Coreutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Perl, og Sed

Påkrevd ved kjøretid: Ingen

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

M4

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Bash og Glibc

Testpakke avhenger av: Diffutils

Må installeres før: Autoconf og Bison

Valgfrie avhengigheter: *libsigsegv*

Make

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Perl og Procps-ng

Må installeres før: Ingen

Valgfrie avhengigheter: *Guile*

Man-DB

Installasjonen avhenger av: Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Sed, og Xz

Påkrevd ved kjøretid: Bash, GDBM, Groff, Glibc, Gzip, Less, Libpipeline, og Zlib

Testpakke avhenger av: Util-linux

Må installeres før: Ingen

Valgfrie avhengigheter: *libseccomp* og *po4a*

Man-Pages

Installasjonen avhenger av: Bash, Coreutils, og Make

Påkrevd ved kjøretid: Ingen

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Meson

Installasjonen avhenger av: Ninja og Python

Påkrevd ved kjøretid: Python

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

MPC

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc, GMP, og MPFR

Testpakke avhenger av: Ingen

Må installeres før: GCC

Valgfrie avhengigheter: Ingen

MPFR

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc og GMP

Testpakke avhenger av: Ingen

Må installeres før: Gawk og GCC

Valgfrie avhengigheter: Ingen

Ncurses

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Bash, GRUB, Inetutils, Less, Procps-ng, Psmisc, Readline, Texinfo, Util-linux, og Vim

Valgfrie avhengigheter: Ingen

Ninja

Installasjonen avhenger av: Binutils, Coreutils, GCC, og Python
Påkrevd ved kjøretid: GCC og Glibc
Testpakke avhenger av: Ingen
Må installeres før: Meson
Valgfrie avhengigheter: *Asciidoc*, *Doxygen*, *Emacs*, og *re2c*

OpenSSL

Installasjonen avhenger av: Binutils, Coreutils, GCC, Make, og Perl
Påkrevd ved kjøretid: Glibc og Perl
Testpakke avhenger av: Ingen
Må installeres før: Coreutils, Kmod, og Linux
Valgfrie avhengigheter: Ingen

Patch

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, og Sed
Påkrevd ved kjøretid: Glibc
Testpakke avhenger av: Diffutils
Må installeres før: Ingen
Valgfrie avhengigheter: *Ed*

Perl

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Groff, Make, Sed, og Zlib
Påkrevd ved kjøretid: GDBM og Glibc
Testpakke avhenger av: Iana-Etc, Less. og Procps-ng
Må installeres før: Autoconf
Valgfrie avhengigheter: *Berkeley DB*

Pkg-config

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, og Sed
Påkrevd ved kjøretid: Glibc
Testpakke avhenger av: Ingen
Må installeres før: Kmod
Valgfrie avhengigheter: *Glib2*

Procps-ng

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Make, og Ncurses

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: DejaGNU

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Psmisc

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, og Sed

Påkrevd ved kjøretid: Glibc og Ncurses

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Python

Installasjonen avhenger av: Bash, Binutils, Coreutils, Expat, GCC, Gdbm, Gettext, Glibc, Grep, Libffi, Make, Ncurses, OpenSSL, Sed, og Util-linux

Påkrevd ved kjøretid: Bzip2, Expat, Gdbm, Glibc, Libffi, Ncurses, OpenSSL, og Zlib

Testpakke avhenger av: GDB og Valgrind

Må installeres før: Ninja

Valgfrie avhengigheter: *Berkeley DB*, *libnsl*, *SQLite*, og *Tk*

Readline

Installasjonen avhenger av: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, og Texinfo

Påkrevd ved kjøretid: Glibc og Ncurses

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Bash, 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

Shadow

Installasjonen avhenger av: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Libcap, Make, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Coreutils

Valgfrie avhengigheter: *CrackLib* og *Linux-PAM*

Sysklogd

Installasjonen avhenger av: Binutils, Coreutils, GCC, Glibc, Make, og Patch

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Sysvinit

Installasjonen avhenger av: Binutils, Coreutils, GCC, Glibc, Make, og Sed

Påkrevd ved kjøretid: Glibc

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Tar

Installasjonen avhenger av: Acl, Attr, Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed, og Texinfo

Påkrevd ved kjøretid: Acl, Attr, Bzip2, Glibc, Gzip, og Xz

Testpakke avhenger av: Autoconf, Diffutils, Findutils, Gawk, og Gzip

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Tcl

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, og Sed

Påkrevd ved kjøretid: Glibc og Zlib

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Texinfo

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, og Sed

Påkrevd ved kjøretid: Glibc og Ncurses

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: Ingen

Util-linux

Installasjonen avhenger av: Bash, Binutils, Coreutils, Diffutils, Eudev, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, og Zlib

Påkrevd ved kjøretid: Glibc, Ncurses, Readline, og Zlib

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: *Libcap-NG*, *Linux-PAM* og *smartmontools*

Vim

Installasjonen avhenger av: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, og Sed

Påkrevd ved kjøretid: Acl, Attr, Glibc, Python, Ncurses, og Tcl

Testpakke avhenger av: Ingen

Må installeres før: Ingen

Valgfrie avhengigheter: *Xorg*, *GTK+2*, *LessTif*, *Ruby*, og *GPM*

wheel

Installasjonen avhenger av: Python

Påkrevd ved kjøretid: Python

Testpakke avhenger av: Ingen testpakke tilgjengelig

Må installeres før: Ingen

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: Eudev, File, GRUB, Kmod, og Man-DB
Valgfrie avhengigheter: Ingen

Zlib

Installasjonen avhenger av: Bash, Binutils, Coreutils, GCC, Glibc, Make, og Sed
Påkrevd ved kjøretid: Glibc
Testpakke avhenger av: Ingen
Må installeres før: File, Kmod, Perl, og Util-linux
Valgfrie avhengigheter: Ingen

Zstd

Installasjonen avhenger av: Binutils, Coreutils, GCC, Glibc, Gzip, Make, og Xz
Påkrevd ved kjøretid: Glibc
Testpakke avhenger av: Ingen
Må installeres før: GCC
Valgfrie avhengigheter: *LZ4*

Appendix D. Oppstarts og sysconfig skriptversjon-20230101

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."
      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

## Screen Dimensions
# Find current screen size
if [ -z "${COLUMNS}" ]; then
   COLUMNS=$(stty size)
   COLUMNS=${COLUMNS##* }
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_deamon 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.

            progname=${program##*/}

            if [[ -e "/run/${progname}.pid" ]]; then
                pidfile="/run/${progname}.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

                    # 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/[^/]*$//'`
            progname=`echo "${program}" | sed "s@${prefix}@@"`

            if [ -e "/run/${progname}.pid" ]; then
                rm -f "/run/${progname}.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 progname
    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 "${prefix}" ]; then
           progname="${program}"
        else
           progname=`echo "${program}" | sed "s@${prefix}@@"`
        fi

        # If a PID file exists with that name, assume that is it.
        if [ -e "/run/${progname}.pid" ]; then
            pidfile="/run/${progname}.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 "${pidfile}"`
        # This can optionally be written as 'sed 1q' to repalce 'head -n1'
        # should LFS move /bin/head to /usr/bin/head
    else
        # Use pidof
        pidlist=`pidof "${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 pidfle] {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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"

    # 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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"

    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: $@ - Message                                                        #
#                                                                             #
# Return values: Not used                                                     #
###############################################################################
log_failure_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"

    # 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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"

    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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${WARNING_PREFIX}${SET_COL}${WARNING_SUFFIX}"

    # 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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SKIP_PREFIX}${SET_COL}${SKIP_SUFFIX}"

    # 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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"

    # 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()
{
    /bin/echo -n -e "${@}"

    # 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
#
# Version     : LFS 7.0
```

```
#
########################################################################

### 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

      (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
#
# Version     : LFS 7.0
#
########################################################################

### 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 stop
      fi

      # Start the udev daemon to continually watch for, and act on,
      # uevents
      /sbin/udevd --daemon

      # Now traverse /sys in order to "coldplug" devices that have
      # already been discovered
      /sbin/udevadm trigger --action=add    --type=subsystems
      /sbin/udevadm trigger --action=add    --type=devices
      /sbin/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 stop
      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="${msg}were found and have been been "
      msg="${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 stop
   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..."

      # As of udev-186, the --run option is no longer valid
      #rundir=$(/sbin/udevadm info --run)
      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 $line ; do
            /sbin/udevadm trigger --subsystem-match=$subsystem --action=add
         done
      done

      # Now wait for udevd to process the uevents we triggered
      if ! is_true "$OMIT_UDEV_RETRY_SETTLE"; then
         /sbin/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/fb0 ] && 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"

      # 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 Burgress (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
#
# 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 kernel and system log daemons.
# Description:         Starts kernel and system log daemons.
#                      /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

      log_info_msg "Starting kernel log daemon..."
      start_daemon /sbin/klogd
      evaluate_retval
      ;;

   stop)
      log_info_msg "Stopping kernel log daemon..."
      killproc /sbin/klogd
      evaluate_retval

      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
      statusproc klogd
      ;;

   *)
      echo "Usage: ${0} {start|stop|reload|restart|status}"
      exit 1
      ;;
esac

exit 0
```

```
# End sysklogd
```

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..."
      killall5 -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..."
      killall5 -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 deliminator 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 - kpfleming@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
   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 - kpfleming@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 - kpfleming@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

   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
```

Appendix E. Udev konfigurasjonsregler

Reglene i dette vedlegget er listet opp for enkelhets skyld. Installasjon gjøres normalt via instruksjoner i Section 8.70, “Eudev-3.2.11”.

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"

# Comms devices

KERNEL=="ippp[0-9]*",       GROUP="dialout"
KERNEL=="isdn[0-9]*",       GROUP="dialout"
KERNEL=="isdnctrl[0-9]*",   GROUP="dialout"
KERNEL=="dcbri[0-9]*",      GROUP="dialout"
```

Appendix 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

Important

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions
   a. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
   b. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
   c. "Licensor" means the individual or entity that offers the Work under the terms of this License.
   d. "Original Author" means the individual or entity who created the Work.
   e. "Work" means the copyrightable work of authorship offered under the terms of this License.
   f. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

g. "License Elements" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, Noncommercial, ShareAlike.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

   a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;

   b. to create and reproduce Derivative Works;

   c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

   d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works;

   The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Sections 4(e) and 4(f).

4. Restrictions.The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

   a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any reference to such Licensor or the Original Author, as requested.

   b. You may distribute, publicly display, publicly perform, or publicly digitally perform a Derivative Work only under the terms of this License, a later version of this License with the same License Elements as this License, or a Creative Commons iCommons license that contains the same License Elements as this License (e.g. Attribution-NonCommercial-ShareAlike 2.0 Japan). You must include a copy of, or the Uniform Resource Identifier for, this License or other license specified in the previous sentence with every copy or phonorecord of each Derivative Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Derivative Works that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder, and You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Derivative Work with any technological measures that control access or use of the Work in a manner

inconsistent with the terms of this License Agreement. The above applies to the Derivative Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Derivative Work itself to be made subject to the terms of this License.

c. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

d. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Derivative Works or Collective Works, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

e. For the avoidance of doubt, where the Work is a musical composition:

   i. Performance Royalties Under Blanket Licenses. Licensor reserves the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work if that performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

   ii. Mechanical Rights and Statutory Royalties. Licensor reserves the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions), if Your distribution of such cover version is primarily intended for or directed toward commercial advantage or private monetary compensation. 6. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

f. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

   a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

   b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

   a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

   b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

   c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

   d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

   e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Important

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, neither party will use the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time.

Creative Commons may be contacted at *http://creativecommons.org/*.

F.2. The MIT License

Index

Packages

Programs

Libraries

Scripts

Others