Linux From Scratch

Version 9.1

Utgiven 1a Mars, 2020

Skapad av Gerard Beekmans
Redaktör: Bruce Dubbs

Linux From Scratch: Version 9.1 : Utgiven 1a Mars, 2020

Skapad av Gerard Beekmans och Redaktör Bruce Dubbs

Innehållsförteckning

Inledning

Förord

Min resa för att lära mig om och bättre förstå Linux tog sin början år 1998. Jag hade just installerat min första Linux-distribution och hade väldigt snart fascinerats av hela konceptet och filosofin bakom Linux.

Det finns alltid många sätt att lösa en uppgift. Detta gäller även Linuxdistributioner. Ett stort antal har genom åren kommit att existera. Vissa existerar ännu, andra har förvandlats till något annat, och ytterligare andra har förpassats till våra minnen. De löser alla saker och ting på olika sätt, för att passa deras användares behov. Eftersom det fanns så många sätt att uppnå samma slutmål, började jag inse att jag inte längre behövde vara begränsad av en enstaka utgåva. Innan det Linux upptäcktes, levde vi med problemen i andra operativsystem eftersom man helt enkelt inte hade något val. Det var vad det var, vare sig du gillade det eller inte. Med Linux, började ett koncept för valfrihet att etableras. Om du inte gillade något var du fri, även uppmuntrad, att göra något åt detta.

Jag provade ett antal distributioner och kunde inte välja en enstaka. De var alla kraftfulla system i sin egen rätt. Det var inte en fråga om rätt eller fel längre. Det hade blivit en fråga om personlig smak. Med alla alternativ tillgängliga blev det uppenbart att det inte skulle gå hitta ett enstaka system som vore perfekt för mig. Så jag påbörjade skapandet av ett eget Linux-system. Ett som helt och hållet skulle vara anpassat till mina personliga preferenser.

För att verkligen göra det till mitt eget system, bestämde jag mig för att kompilera allt från källkod, istället för att använda förkompilerade binära paket. Detta “perfekta” Linux-system skulle implementera styrkorna från diverse system, utan att lida av deras upplevda svagheter. Till en början, var idén relativt svindlande. Jag bibehöll övertygelsen att det var möjligt.

Efter att ha sorterat igenom problem såsom cirkulära beroenden och compile-time errors, slutförde jag till sist mitt Linux-system. Det var helt och hållet användbart och av samma kvalitet som andra dåtida distributioner. Men det var min egen skapelse. Det var väldigt tillfredställande att själv ha satt ihop ett sådant system. Det enda som hade varit bättre vore om jag skrivit all kod själv. Detta var det näst bästa alternativet.

Allteftersom jag delade mina mål och erfarenheter med andra medlemmar av Linux-gemenskapen, blev det tydligt att det fanns ett ihållande intresse för dessa idéer. Det blev även tydligt att sådana ursprungs-byggda system inte endast erbjöd specifika användarmöjligheter, utan även innebar en möjlighet för Linux-användare att utöka sina kunskaper om hur systemen vilka de använde fungerade. Det var till följd av detta som projektet *Linux From Scratch* tog sin början.

Denna bok är detta projekts centrala kärna. Den tillhandahåller bakgrunden, samt de nödvändiga instruktionerna för att bygga ditt eget system. Medan denna bok tillhandahåller en mall vilken resulterar i ett korrekt fungerande system, så är du fri att modifiera instruktionerna för att passa dig själv, vilket även är, till viss del, en viktig aspekt av detta projekt. Du kvarstår i kontroll; vi erbjuder endast en hjälpande hand att leda dig ut på din egen resa.

Det är min genuina önskan att du kommer uppskatta upplevelsen av att jobba på detta projekt, och skapar ett Linux-system som är ditt eget verk.

--
Gerard Beekmans
gerard@linuxfromscratch.org

Målgrupp

Det finns många anledningar till varför någon skulle vilja läsa denna bok. En av de frågor många ställer är “varför gå igenom alla problem med att manuellt bygga ett Linux-system från grunden, när man bara kan ladda ned och installera ett redan existerande?”.

En viktig anledning till varför detta projekt existerar, är för att lära dig hur ett Linux-system fungerar på djupet. Att bygga ett LFS-system demonstrerar vad som driver Linux, och hur saker samverkar och beror på varandra. En av de främsta fördelarna som denna lärupplevelse kan bidra med är förmågan att anpassa ett Linuxsystem till dina egna unika behov.

En annan betydelsefull fördel, är att LFS möjliggör för dig att utöva mer omfattande kontroll över ditt system, samt att du inte blir beroende av någon annans distribution. Med LFS dikterar du villkoren för hela systemet.

LFS möjliggör även skapandet av ett väldigt kompakt Linux-system. Då man installerar vanliga distributioner, blir man ofta tvingad till att installera mångra program som ofta inte används. Dessa program slösar på resurser. Man kan dock argumentera för att det inte spelar så stor roll längre, till följd av dagens kraftfulla datorer. Oavsett detta, så är man ibland begränsad vad gäller resurser, och då vanligtvis vad gäller lagring. Föreställ dig att man önskar installera ett system på exempelvis ett USB eller ett inbyggt system. Detta är tillfällen då LFS kan vara mycket fördelaktigt.

Ytterligare en fördel med ett system byggt från grunden är säkerhet. Genom att kompilera hela systemet från källkod är det möjligt för dig att kontrollera alla paket och implementera alla säkerhetspatcher som du anser vara relevanta. Du behöver inte längre vänta på att någon annan ska täppa till ett säkerhetshål. Om du inte kontrollerar och implementerar säkerhetspatchen själv, vet du inte heller om den är korrekt byggd och framgångsrikt löser problemet.

Målet med LFS är att bygga ett komplett och funktionellt system på grundläggande nivå. Om du inte önskar bygga ditt eget Linuxsystem från grunden, är det i vilket fall möjligt att du kan lära dig mycket från informationen i denna bok.

Det finns för många anledningar till varför man borde bygga sitt eget Linuxsystem för att räkna upp dem här. Men om den absolut viktigaste anledningen ska identifieras, så är det lärupplevelsen i sig.

LFS mål-arkitekturer

De huvudsakliga målarkitekturerna för LFS är AMD/Intel x86 (32-bit) och x86_64 (64-bit) CPUer. Å andra sidan, är instruktionerna i denna bok också, med några modifikationer, kompatibla med Power PC och ARM CPUer.
För att bygga ett system som använder en av dessa CPUer, är det huvudsakliga kravet att ha ett tidigare Linux-system vilket är anpassat för den arkitektur vilken du ska bygga. Notera även att en 32-bits distribution kan installeras och användas som ett värdsystem på en 64-bit AMD/Intel dator.

För att bygga LFS, är fördelen med ett 64-bit system genetmot ett 32-bit minimal. För ett testsystem av LFS 9.1 byggt för ett Core i7-4790 baserat system, vilket använder sig av 4 kärnor, uppmättes följande mått:

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

Som du kan se, är 64-bit systemet, på samma hårdvara, endast 3% snabbare men 22% större än 32-bits varianten. Om du planerar att använda LFS som en LAMP-server, eller brandvägg, är ett 32-bit system tillräckligt. Å andra sidan, flertalet av paketen i BLFS behöver nu mer än 4GB ram för att byggas och/eller köras, så ifall du planerar att att använda LFS som standard desktopsystem, rekommenderar författarna ett 64-bit system.

Det standard 64-bit system vilket är resultatet av LFS, betraktas som ett “rent” 64-bit system. Detta innebär att det endast stödjer 64-bit program. Att bygga ett “multi-lib” system kräver att flera applikationer måste kompileras två gånger. En gång för 32-bit, och en gång för 64-bit. Detta är något som inte direkt stödjs av LFS eftersom det skulle störa målet att LFS ska tillhandahålla instruktionerna för det rättframma byggandet av ett grundläggande Linux-system. Det finns forks av LFS för multilib, vilka är tillgängliga på http://www.linuxfromscratch.org/~thomas/multilib/index.html, men det är en avancerad aspekt att ge sig in på och därför något vi utelämnar i denna bok.

Förkrav

Att bygga ett LFS-system är inte en enkel uppgift. Det förutsätter en viss nivå av tidigare kunskap vad gäller administration av Linuxsystem, för att man korrekt ska kunna lösa problem samt utföra de listade kommandona. Du borde, som ett absolut minimum, ha förmågan att använda kommando-skalet för att kopiera, flytta, röra dig emellan samt lista filer och mappar. Du förväntas också ha en relativt grundläggande erfarenhet av att installera och använda Linuxprogram.

Eftersom denna bok utgår ifrån *åtminstone* denna kunskap, kommer antagligen inte de diverse LFS-forumen att bistå med hjälp vad gäller problem inom dessa områden. Du kommer upptäcka att sådana grundläggande frågor inte kommer besvaras, alternativt att du hänvisas till den läslista för LFS som är etablerad.

Innan det att du påbörjar byggandet av ditt eget LFS-system, rekommenderar vi att du läser igenom följande:

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

  Detta är en omfattande guide vad gäller att bygga och installera “generiska” Unix-mjukvaru paket, under Linux. Även om den skrevs för ett bra tag sedan, tillhandahåller den ännu en god sammanfattning vad gäller de tekniker vilka används för att bygga och installera mjukvara.
- Beginner's Guide to Installing from Source *http://moi.vonos.net/linux/beginners-installing-from-source/*

  Denna guide tillhandahåller en god översikt vad gäller grundläggande tekniker för att bygga mjukvara från källkod.

LFS och Standarder

Strukturen för LFS representerar diverse Linux standarder så gott som möjligt. De primära standarderna är:

- *POSIX.1-2008*.
- *Filesystem Hierarchy Standard (FHS) Version 3.0*
- *Linux Standard Base (LSB) Version 5.0 (2015)*

LSB har fyra separata standarder: Core, Desktop, Runtime Languages, och Imaging. Utöver generiska krav existerar även arkitektur-specifika krav. Det existerar även två områden för test användning: Gtk3 samt Graphics. LFS efterstävar att vara anpassat till de arkitekturer vilka diskuterades i den föregående sektionen.

Kommentar

Många människor håller inte med om kraven för LSB. Den huvudsakliga anledning till att definiera dessa är att säkerställa att proprietär mjukvara kan installeras och drivas som tänkt. Eftersom LFS är källkods-baserat, har användaren total kontroll över vilka paket som är önskvärda, och till följd av detta väljer många att inte installera vissa paket vilka specifieras av LSB.

Att skapa ett komplett LFS-system kapabelt att uppfylla kraven för LSBs certifierings-test är möjligt, men inte utan att först installera många extra paket vilket inte är inkluderade i LFS. Dessa paket återfinns istället i BLFS.

Paket tillhandahållna av LFS, nödvändiga för att tillfredställa LSB-kraven.

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

LSB Runtime Languages:	Perl
LSB Imaging:	Inga
LSB Gtk3 and LSB Graphics (Trial Use):	Inga

Paket tillhandahållna av BLFS, nödvändiga för att tillfredställa LSB-kraven.

LSB Core:	At, Batch (en del av At), Cpio, Ed, Fcrontab, LSB-Tools, NSPR, NSS, PAM, Pax, Sendmail (eller Postfix eller Exim), time
LSB Desktop:	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 Runtime Languages:	Python, Libxml2, Libxslt
LSB Imaging:	CUPS, Cups-filters, Ghostscript, SANE
LSB Gtk3 and LSB Graphics (Trial Use):	GTK+3

Paket *EJ* tillhandahållna av LFS eller BLFS, nödvändiga för att tillfredställa LSB-kraven.

LSB Core:	Inga
LSB Desktop:	Qt4 (men Qt5 tillhandahålls)
LSB Runtime Languages:	Inga
LSB Imaging:	Inga
LSB Gtk3 and LSB Graphics (Trial Use):	Inga

Logisk grund för paket i denna bok

Som tidigare nämnt, är målet med LFS att bygga ett komplett system på grundnivå. Detta inkluderar alla paket vilka är nödvändiga för att systemet ska kunna replikera sig självt, samtidigt som det tillhandahåller en relativt minimal bas från vilken att anpassa ett mer komplett system baserat på användarens önskemål. Detta innebär inte att LFS är det minsta systemet möjligt. Flertalet viktiga paket inkluderas, utan att för den skull vara nödvändiga. Orsaken till att de olika paketen valts återfinns nedan:

- Acl

  Innehåller funktionalitet för administration av Access Control Lists, vilka används för mer detaljerade begränsningar av åtkomst till filer och mappar.

- Attr

  Innehåller program för att administrera utvidgade attribut för filsystemsobjekt.

- Autoconf

  Innehåller program för att producera skript vilka automatiskt kan konfigurera källkod från en utvecklares mall. Det är ofta nödvändigt att bygga om ett paket efter uppdateringar till byggprocessen.

- Automake

  Innehåller program för att generera Make-filer från mallar. Det är ofta nödvändigt att bygga om ett paket efter uppdateringar till byggprocessen.

- Bash

Tillfredställer ett av LSBs kärn-krav: Att tillhandahålla ett Bourne Shell Interface för systemet. Det valdes över andra paket eftersom dess generalla användningsområden och funktionalitet är mer omfattande än det hos andra skal.

- Bc

  Tillhandahåller ett arbitrary precision numeric processing språk. Det tillfredställer ett krav som LSB ställer då Linux kernel byggs.

- Binutils

  Innehåller en länkare, en assembler, samt andra verktyg nödvändiga för att hantera objektfiler. Programmen i detta paket är nödvändiga för att kompilera de flesta av paketen i ett LFS-system.

- Bison

  Innehåller GNU-versionen av yacc(Yet Another Compiler Compiler). Yacc krävs för att det ska gå att bygga flertalet övriga LFS-program.

- Bzip2

  Innehåller program för att komprimera samt dekomprimera filer. Det krävs för att det ska gå att dekomprimera flertalet av LFS-programmen.

- Check

  Innehåller en test-sele(Test Harness) som andra program kan använda sig av.

- Coreutils

  Innehåller ett antal program vilka krävs för att det ska gå att visa och manipulera filer och mappar. Dessa program används för kommandolinje-baserad filhantering, och är nödvändiga för installationsproceduren av varje enstaka LFS -program.

- DejaGNU

  Innehåller ett ramverk för att testa andra program. Det installeras endast i den temporära toolchainen.

- Diffutils

  Innehåller program som specifierar skillnaden emellan filer eller mappar. Dessa program kan användas för att skapa patchar, och används under många andra pakets byggprocedurer.

- E2fsprogs

  Innehåller funktionalitet för att hantera ext2, ext3 samt ext4 filsystem. Dessa är de vanligaste samt mest omfattnade testade filsystemen för Linux.

- Eudev

  Detta paket innehåller en enhetshanterare. Det kontrollerar dynamiskt innehållet i mappen /dev för att upptäcka då enheter läggs till eller tas bort från systemet.

- Expat

  Innehåller ett relativt litet XML-bibliotek. Det krävs för att XML::Parser Perl modulen skall fungera som det är tänkt.

- Expect

  Innehåller ett program för att utföra skriptade dialoger med andra interaktiva program. Det används generellt sätt för testning av andra paket. Det är endast installerat i den temporära tool-chainen.

- File

Innehåller funktionalitet för att avgöra typen av en eller flera givna filer. Några paket behöver det för sin byggprocess.

- Findutils

  Innehåller program för att hitta filer i ett filsystem. Det används av många pakets byggskript.

- Flex

  Innehåller ett verktyg för att genera program vilka analyserar mönster i text. Det är GNU-versionen av programmet lex(lexical analyzer). Det behövs för att bygga flertalet LFS-paket.

- Gawk

  Innehåller program för att manipulera textfiler. Det är GNU-versionen av awk(Aho-Weinberg-Kemighan). Det används i många andra programs byggskript.

- Gcc

  Innehåller GNU Compiler Collection. Det innehåller C samt C++ kompilatorerna, samt diverse andra kompilatorer vilka inte används av LFS.

- GDBM

  Innehåller GNU Database Manager Library. Det används av ett annat LFS-program: Man-DB.

- Gettext

  Innehåller verktyg och bibliotek för internalisering och lokalisering av flertalet paket.

- Glibc

  Innehåller det huvudsakliga C-biblioteket. Linux-program fungerar inte utan det bibliotek.

- GMP

  Innehåller matematikbibliotek vilka tillhandahåller nödvändiga funktioner för arbitrary precision arithmetic. För att bygga GCC är det absolut nödvändigt att dessa bibliotek finns tillgängliga.

- Gperf

  Innehåller ett program som genererar en perfekt hash-funktion från ett nyckel-set. Krävs av Eudev.

- Grep

  Innehåller program för att söka igenom filer. Dessa program används av de flesta pakets byggskript.

- Groff

  Innehåller program för att processa och formatera text. Dessa program används av de flesta pakets bygg-skript.

- GRUB

  Detta paket innehåller Grand Unified Boot Loader. Det finns fler bootloaders, men denna är den mest flexibla.

- Gzip

  Innehåller program för att komprimera och dekomprimera filer. Det behövs för att dekomprimera många paket vilka används i LFS.

- Iana-etc

Tillhandahåller data för nätverkstjänster samt protokoll. Det behövs för att möjliggöra den förväntade nätverks-funktionaliteten.

- Inetutils

  Innehåller program för grundläggande nätverksadministration.

- Intltool

  Innehåller verktyg för att extrahera översättningsbara strängar från käll-filer.

- IProute2

  Innehåller program för grundläggande och avancerade IPv4 och IPv6 nätverk. Det valdes över de andra traditionella paketen med nätverksverktyg, på grund av sina IPv6 funktionaliteter.

- Kbd

  Innehåller key-table filer, tangentbords funktionalitet för andra sådana än Amerikanska, samt divers konsolfonter.

- Kmod

  Innehåller program vilka behövs för att det ska gå att administrera Linux kernelmoduler.

- Less

  Innehåller ett program för att scrolla upp och ned genom en textfil. Det används även av Man-DB.

- Libcap

  Implementerar användarens gränssnitt enligt POSIX 1003. Alltså funktionalitet tillgänglig inuti Linux kernel.

- Libelf

  Tillhandahåller bibliotek och verktyg för ELF-filer och DWARF-data. De flesta verktyg i detta paket är tillgängliga även i andra paket, men biblioteket krävs för att bygga Linux kernel enligt den standardiserade(och mest effektiva) konfigurationen.

- Libffi

  Implementerar ett portabelt, högnivå programmerings gränssnitt för diverse anropande konventioner. Vissa program kanske under kompilationstid inte vet vilka argument som används för att anropa en funktion. Exempelvis en tolk kan bli instruerad vid run-time vad gäller antal och typer av argument som används för att anropa en viss funktion. Libffi kan då användas för att tillhandahålla ett sorts bro emellan tolk och kompilerad kod.

- Libpipeline

  Innehåller ett bibliotek för att manipulera pipelines av subprocesser, på ett flexibelt och relevant vis. Det krävs av Man-db paketet.

- Libtool

  Innehåller GNUs generiska bibliotek support skript. Det wrapar komplexiteten av att använda delade bibliotek, till ett sammanhängade och portabelt gränssnitt. Det krävs av test-sviterna i LFS.

- Linux Kernel

  Detta paket är operativsystemet självt. Det är vad som gör LFS till Linux.

- M4

Innehåller en generell makro processor, vilken används om ett byggverktyg av andra program.

- Make

  Innehåller ett program för att dirigera byggandet av paket. Det krävs av nästan alla paket i LFS.

- Man-DB

  Innehåller ett program för att hitta och visa man sidor. Det valdes över paketet Man, till följd av dess överlägsna förmåga till internalisering. Det tillhandahåller även paketet Man.

- Man-pages

  Innehåller Man sidorna för grundläggande Linux.

- Meson

  Tillhandahåller ett mjukvaruverktyg för att automatisera byggandet av mjukvara. Det huvudsakliga målet för Meson är att minimisera den tid det tar för utvecklare att bygga system.

- MPC

  Innehåller funktioner för aritmetik av komplexa nummer. Används av GCC.

- MPFR

  Innehåller funktioner för multiple precision arithmetic. Används av GCC.

- Ninja

  Innehåller ett mindre byggsystem optimerat för hastighet. Det är designat för att få sina inmatade filer tillhandahållna av ett byggsystem på en högre nivå, och att sedan bygga dessa filer så snabbt som möjligt.

- Ncurses

  Innehåller bibliotek för terminal-självständing hantering av character screens. Det används ofta för att tillhandahålla kontroll för t.ex ett menysystem. Det krävs av ett antal LFS-paket.

- Openssl

  Tillhandahåller hanteringsverktyg och bibliotek relaterade till kryptografi. Dessa är användbara för att tillhandahålla kryptografiska funktioner till andra paket, exempelvis till Linux kernel.

- Patch

  Innehåller ett program vilket kan modifiera eller skapa filer genom att tillämpa en *patch*-fil vanligen skapad av diff. Det krävs för att bygga flertalet LFS-paket.

- Perl

  En tolk för run-time språket PERL. Det krävs för installationen av flera LFS-paket, samt för körandet av deras test-sviter.

- Pkg-config

  Tillhandahåller ett program vilket returnerar metadata relaterat till ett installerat bibliotek eller paket.

- Procps-NG

  Innehåller program för att monitorera processer. Användbara för systemadministration, och används även av LFS boot-skripts.

- Psmisc

  Innehåller program vilka visar information om processer vilka körs. Dessa är användbara för systemadministration.

- Python 3

  Innehåller Python 3, vilket är ett språk där läsbarhet betonas som en viktig aspekt av kodning.

- Readline

  Ett antal bibliotek som tillhandahåller editering och historiesökning av kommandolinjen. Används av Bash.

- Sed

  Möjliggör editering av textfiler utan att de behöver öppnas i en texteditor. Det krävs även av de flesta av LFS konfigurationsskript.

- Shadow

  Innehåller program för att hantera lösenord på ett säkert sätt och vis.

- Sysklogd

  Innehåller program vilka loggar systemmeddelanden, likt de vilka kommer ifrån kernel eller daemonprocesser.

- Sysvinit

  Tillhandahåller init-programmet, vilket är föräldern till alla andra processer på ett Linux-system.

- Tar

  Tillhandahåller funktionalitet för arkivering och extrahering av mer eller mindre alla paket använda i LFS.

- Tcl

  Innehåller Tool Command Language, vilket används i många test-sviter i LFS. Installeras endast i den temporära toolchainen.

- Texinfo

  Innehåller program för att läsa, skriva, samt konvertera info sidor. Det används under installationsproceduren för många LFS-paket.

- Util-linux

  Innehåller diverse verktygsprogram. Ibland dem finns verktyg för att hantera filsystem, konsoler, partitioner, samt meddelanden.

- Vim

  Innehåller en texteditor. Valdes till följd av dess kompatibilitet med den klassiska Vi-editorn, samt dess stora mängd av kraftfulla funktioner.

- XML::Parser

  En Perl-modul med gränsnitt till Expat.

- XZ Utils

  Innehåller program för komprimering samt dekomprimering av filer. Det möjliggör den mest effektiva komprimeringen som generellt sett finns tillgänglig och är användbart om man önskar dekomprimera paket i XZ eller LZMA format.

- Zlib

  Innehåller komprimerings och dekomprimerings rutiner använda av ett antal andra program.

- Zstd

  Innehåller komprimerings och dekomprimerings rutiner använda av ett antal andra program. Det möjliggör hög komprimeringsratio och en väldigt omfattande valbarhet vad gäller komprimering vs hastighet.

Typografi (I denna bok återanvänds inte denna standard helt och hållet)

För att underlätta saker och ting så används i denna bok ett antal typografiska konventioner. Denna sektion innehåller ett antal exempel avseende det typografiska format som återfinns i Linux From Scratch.

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

Denna typ av text är tänkt att skrivas exakt så som ovan, bortsett ifrån något annat nämns i den omgivande texten. Den används också i förklaringstexten för att identifiera vilket av kommandona som refereras.

I vissa fall är en logisk rad utvidgad till två eller fler fysiska linjer, och markeras då med ett backslash i slutet av linjen.

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

Notera att backslashet måste följas av ett omedelbart return. Andra whitespaces leder till inkorrekta resultat.

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

Denna form av text(fixerad-bredd text) visar skärmens output, vanligtvis som ett resultat av kommandon. Detta format visar även filnamn, såsom t.ex: `/etc/ld.so.conf`.

Emphasis

Denna form av text används i boken för ett flertal syften. Dess huvudsakliga syfte är att betona viktiga poänger/aspekter.

http://www.linuxfromscratch.org/

Detta format används för att betona hyperlänkar, både inom projektet LFS samt till externa sidor.

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

Detta format används då konfigurationsfiler skapas. Första kommandot instruerar systemet att skapa filen `$LFS/etc/group` från vad som än skrivs på linjer vilka tills sekvensen End Of File(EOF) nås. Till följd av detta, representeras denna sektion vanligtvis likt ovan,

`<REPLACED TEXT>`

Detta format används för att enkapsulera text vilken inte är att tänkt att skrivas som representerad, eller för copy-paste.

`[OPTIONAL TEXT]`

Detta format används för att enkapsulera text som är valbar.

`passwd(5)`

Detta format används för att referera till en specifik Man(#) sida. Numret inom parentes indikerar en specifik sektion i manualen. Exempelvis **Passwd** har två Man sidor. För att komma till den korrekta sidan skriver man därför:
man 5 passwd
Rent generellt är det dock fullt tillräckligt att skriva **man <program name>** för att hitta den korrekt informationen.

Struktur

Denna bok är uppdelad i följande sektioner

Del I – Introduktion

Del I går igenom ett antal viktiga aspekter vad gäller hur man bör fortfölja med installationen av LFS. Denna sektion innehåller även metadata vad gäller boken.

Del II – Förberedelser för byggnation

Del II beskriver vilka förberedelser som krävs för att bygga systemet – skapandet av en partition; nedladdning av paket; kompilering av temporära verktyg.

Del III – Att bygga LFS-systemet

Del III guidar läsaren genom byggnationen av LFS-systemet – kompilering och installation av alla paket ett efter ett; Implementering av bootskripts; installation av kernel. Det resulterande Linuxsystemet är fundamentet på vilket annan mjukvara sedan kan byggas för att expandera systemet på ett önskat sätt ochvis. Vid slutet av boken återfinns en lista vilken refererar alla program, bibliotek, samt viktiga filer vilka är tänkta att ha installerats.

Errata

Mjukvaran vilken används för att skapa ett LFS-system uppdateras och vidareutvecklas konstant. Säkerhetsvarningar samt buggfixar kan bli tillgängliga efter att boken publicerats. För att dubbelkolla ifall paket-versioner eller instruktioner i denna release är i behov av modifikationer, var vänlig besök:
http://www.linuxfromscratch.org/lfs/errata/9.1/
Gör gärna detta innan du fortgår med byggandet av ditt system, och notera vilka anpassningar som krävs samt hur och var de ska implementeras.

Del I. Introduktion

Kapitel 1. Introduktion

1.1. Hur man bygger ett LFS-system

LFS-systemet kommer byggas med hjälp av ett redan installerat Linuxsystem(exempelvis Debian, OpenMandriva, Fedora). Detta existerande system(värden) kommer att användas som startpunkten för att tillhandahålla det nya systemets nödvändiga program(däribland kompilator, länkare, kommandoskal). Markera valet “development” då distributionen installeras, för att få tillgång till dessa verktyg.

Alternativt, ifall du inte önskar installera ett extra system på din dator, är det möjligt att använda en LiveCD/USB från en kommersiell distribution.

Kapitel 2 i denna bok beskriver hur man skapar en Linux native partition samt eth filsystem. Detta är den plats där det nya LFS-systemet kommer att kompileras samt installeras. Kapitel 3 förklarar vilka paket och patchar som behöver laddas ned för att bygga systemet, samt hur de ska förvaras på det nya filsystemet. Kapitel 4 diskuterar etablerandet av en lämplig systemmiljö. Var vänlig läs kapitel 4 noggrant, då det behandlar ett flertal viktiga aspekter vilka du behöver vara medveten om innan det att du - i kapitel 5 - påbörjar själva byggnationen.

Kapitel 5 går igenom installationen av ett antal paket vilka kommer leda till den grundläggande utvecklingssviten (toolchainen), vilken i kapitel 6 kommer att användas för att bygga själva systemet. Vissa av dessa paket behövs för att det ska vara möjligt att uppfylla kraven för cirkulära beroenden – för att kompilera en kompilator, behövs en kompilator.

Kapitel 5 går även igenom hur du bygger en första version av din toolchain, innehållande Binutils och GCC(“första version” innebär att dessa två paket sedan ominstalleras). Nästa steg är att bygga Glibc, C-biblioteket. Glibc kommer att kompileras med hjälp av version #1 av toolchainen. Efter detta kommer en version #2 att byggas. Denna gång så kommer toolchainen att dynamiskt länkas gentemot det nya Glibc. De kvarvarande paketen i kapitel 5 kommer att byggas med hjälp av toolchain #2. När detta väl är gjort, kommer LFS-systemet inte längre vara beroende av värdsystemet, med undantag för dess kernel.

Denna ansträngning att isolera LFS-systemet från värdsystemet kan verka överdriven. En teknisk förklaring till varför detta ändå görs, återfinns i sektionen 5.2, “Toolchain tekniska kommentarer”.

I kapitel 6 så byggs det kompletta LFS-systemet. Programmet **chroot** (change root) används för att träda in i en virtuell miljö och skapa ett nytt skal vilkets root-mapp kommer sättas till LFS-partitionen. Detta liknar processen att reboota och instruera kerneln att montera LFS-partitionen som root-partition. I detta fall rebootar inte systemet, utan använder istället **chroot** eftersom det innebär arbete, vilket tillsvidare inte är nödvändigt, för att skapa ett bootbart system. Att chrooting möjliggör att man fortsatt kan använda värd-systemet medans LFS-systemet byggs, är den huvudsakliga anledningen. Medan det att du kompilerar, kan du fortfarande använda din dator som vanligt.

För att slutföra installationen, så återges i kapitel 7 systemets grundläggande konfigurationer. Kernel och bootloader etableras i kapitel 8. Kapitel 9 innehåller information för den som önskar att fortsätta LFS-projektet. Oavsett: Efter det att alla steg i denna bok har gåtts igenom, kommer det vara möjligt att boota in i ditt nya LFS-system.

Detta är processen i ett nötskal. Detaljerad information vad gäller varje steg återfinns i de följande kapitlen och paket-beskrivningarna. Aspekter vilka tycks krångliga att förstå kommer förklaras, och allt eftersom du går vidare med ditt projekt kommer saker och ting att falla på plats.

1.2. Nytt sedan senaste releasen

Nedan återfinns en lista över vilka förändringar som gjorts sedan den förra releasen av boken.

Uppgraderade till:

-
- Bc 2.5.3
- Binutils-2.34
- Bison-3.5.2
- Check-0.14.0
- E2fsprogs-1.45.5
- Eudev-3.2.9
- Expat-2.2.9
- File-5.38
- Findutils-4.7.0
- Glibc-2.31
- GMP-6.2.0
- Grep-3.4
- IPRoute2-5.5.0
- Libcap-2.31
- Libelf-0.178 (from elfutils)
- Libffi-3.3
- Libpipeline-1.5.2
- Linux-5.5.3
- Make-4.3
- Man-DB-2.9.0
- Man-pages-5.05
- Meson-0.53.1
- Ncurses-6.2
- Ninja-1.10.0
- Openssl-1.1.1d
- Perl-5.30.1
- Python-3.8.1
- Sed-4.8
- Shadow-4.8.1
- SysVinit-2.96
- Tcl-8.6.10
- Texinfo-6.7

- Tzdata-2019c
- Util-Linux-2.35.1
- Vim-8.2.0190

Tillagt:

- Zstd-1.4.4

Borttaget:

-

1.3. Changelog

This is version 9.1 of the Linux From Scratch book, dated March 1st, 2020. If this book is more than six months old, a newer and better version is probably already available. To find out, please check one of the mirrors via *http://www.linuxfromscratch.org/mirrors.html*.

Below is a list of changes made since the previous release of the book.

Changelog Entries:

- 2020-03-01
  - [bdubbs] - LFS-9.1 released.
- 2020-02-14
  - [bdubbs] - Uppdaterad till bison-3.5.2. Fixes *#4597*.
- 2020-02-13
  - [bdubbs] - Uppdaterad till ncurses-6.2. Fixes *#4596*.
  - [bdubbs] - Uppdaterad till man-pages-5.05. Fixes *#4595*.
  - [bdubbs] - Uppdaterad till linux-5.5.3.tar.xz. Fixes *#4592*.
- 2020-01-27
  - [bdubbs] - Uppdaterad till vim-8.2.0190. Addresses *#4500*.
  - [bdubbs] - Uppdaterad till binutils-2.34. Fixes *#4590*.
  - [bdubbs] - Uppdaterad till glibc-2.31. Fixes *#4589*.
  - [bdubbs] - Uppdaterad till linux-5.5.1. Fixes *#4588*.
  - [bdubbs] - Uppdaterad till bc-2.5.3. Fixes *#4587*.
  - [bdubbs] - Uppdaterad till iproute2-5.5.0. Fixes *#4586*.
  - [bdubbs] - Uppdaterad till util-linux 2.35.1. Fixes *#4560*.
- 2020-01-27
  - [bdubbs] - Uppdaterad till ninja-1.10.0. Fixes *#4585*.
  - [bdubbs] - Uppdaterad till check-0.14.0. Fixes *#4583*.
  - [bdubbs] - Uppdaterad till shadow-4.8.1. Fixes *#4582*.
  - [bdubbs] - Uppdaterad till meson-0.53.1. Fixes *#4581*.

- [bdubbs] - Uppdaterad till linux-5.5. Fixes *#4580*.
- [bdubbs] - Uppdaterad till bison-3.5.1. Fixes *#4579*.

- 2020-01-19
  - [bdubbs] - Uppdaterad till make-4.3. Fixes *#4578*.
  - [bdubbs] - Uppdaterad till vim-8.2.0129. Addresses *#4500*.
  - [bdubbs] - Uppdaterad till gmp-6.2.0. Fixes *#4577*.
  - [bdubbs] - Uppdaterad till sed-4.8. Fixes *#4576*.
  - [bdubbs] - Uppdaterad till bc-2.5.1. Fixes *#4575*.
  - [bdubbs] - Uppdaterad till linux-5.4.13. Fixes *#4572*.
- 2020-01-16
  - [pierre] - Uppdaterad till libcap-2.31. Fixes *#4574*.
- 2020-01-13
  - [bdubbs] – Säkerställ att zstd libraries är installerade i den korrekta platsen.
- 2020-01-12
  - [bdubbs] – Lade till zstd-1.4.4.
- 2020-01-09
  - [bdubbs] - Uppdaterad till meson-0.53.0. Fixes *#4571*.
  - [bdubbs] - Uppdaterad till e2fsprogs-1.45.5. Fixes *#4570*.
  - [bdubbs] - Uppdaterad till grep-3.4. Fixes *#4568*.
  - [bdubbs] - Uppdaterad till libpipeline-1.5.2. Fixes *#4567*.
  - [bdubbs] - Uppdaterad till linux-5.4.8. Fixes *#4566*.
  - [pierre] – Add `/etc/os-release` till sysV versionen, då vissa paket i BLFS kräver detta.
- 2020-01-06
  - [pierre] - Uppdaterad till libcap-2.30. Fixes *#4569*.
- 2020-01-04
  - [pierre] - Fixade diverse issues i libcap-2.29, och uppdaterade beroenden.
- 2020-01-01
  - [bdubbs] - Uppdaterad till libcap-2.29. Fixes *#4564*.
- 2019-12-22
  - [pierre] - Uppdaterad till python3-3.8.1. Fixes *#4564*.
  - [pierre] - Uppdaterad till file-5.38. Fixes *#4563*.
  - [pierre] - Uppdaterad till linux-5.4.6. Fixes *#4562*.
  - [pierre] - Uppdaterad till vim-8.2.0024. Part of *#4500*.
- 2019-12-12
  - [bdubbs] - Uppdaterad till libcap-2.28. Fixes *#4559*

- [bdubbs] - Uppdaterad till bison-3.5. Fixes *#4561*.

- 2019-12-10
  - [renodr] - Fixade en regression i meson som skapade problem med fristående paket.
- 2019-12-05
  - [renodr] - Uppdaterad till bc-2.4.0. Fixes *#4556*.
  - [renodr] - Uppdaterad till shadow-4.8. Fixes *#4557*.
  - [renodr] - Uppdaterad till linux-5.4.2. Fixes *#4558*.
- 2019-12-01
  - [bdubbs] – Lade till upstream fixes patch för bash. Lade även till kommentar gällande problem då man byter till LFS användaren.
  - [bdubbs] - Uppdaterad till vim-8.1.2361. Updates *#4500*.
  - [bdubbs] - Uppdaterad till meson-0.52.1. Fixes *#4555*.
  - [bdubbs] - Uppdaterad till elfutils-0.178. Fixes *#4553*.
  - [bdubbs] - Uppdaterad till iproute2-5.4.0. Fixes *#4551*.
  - [bdubbs] - Uppdaterad till libffi-3.3. Fixes *#4550*.
  - [bdubbs] - Uppdaterad till tcl-8.6.10. Fixes *#4549*.
  - [bdubbs] - Uppdaterad till man-pages-5.04. Fixes *#4548*.
  - [bdubbs] - Uppdaterad till perl-5.30.1. Fixes *#4547*.
  - [bdubbs] - Uppdaterad till linux-5.4.1. Fixes *#4546*.
  - [bdubbs] - Uppdaterad till bc-2.3.2. Fixes *#4545*.
- 2019-11-08
  - [renodr] - Uppdaterad till Linux-5.3.9. Resolves a regression with restarting systems with HD Audio (hda), data corruption on btrfs, and a security vulnerability with systems that use the RTLWIFI driver. Fixes *#4544*.
- 2019-10-31
  - [dj] - Uppdaterad till lfs-bootscripts-20191031.
- 2019-10-25
  - [dj] - Uppdaterad till lfs-bootscripts-20191025.
- 2019-11-01
  - [bdubbs] - Uppdaterad till linux-5.3.8. Fixes *#4539*.
  - [bdubbs] - Uppdaterad till bc-2.2.0. Fixes *#4543*.
  - [bdubbs] - Uppdaterad till check-0.13.0. Fixes *#4540*.
  - [bdubbs] - Uppdaterad till eudev-3.2.9. Fixes *#4542*.
  - [bdubbs] - Uppdaterad till man-db-2.9.0. Fixes *#4541*.
- 2019-10-17
  - [bdubbs] - Flyttar attr och acl till att vara framför Shadow.
  - [bdubbs] - Uppdaterad till linux-5.3.6. Fixes *#4534*.

- [bdubbs] - Uppdaterad till man-pages-5.03. Fixes *#4536*.
- [bdubbs] - Uppdaterad till meson-0.52.0. Fixes *#4535*.
- [bdubbs] - Uppdaterad till Python-3.8.0. Fixes *#4538*.
- [bdubbs] - Uppdaterad till binutils-2.33.1. Fixes *#4537*.

- 2019-09-29
  - [bdubbs] - Uppdaterad till texinfo-6.7. Fixes *#4529*.
  - [bdubbs] - Uppdaterad till e2fsprogs-1.45.4. Fixes *#4530*.
  - [bdubbs] - Uppdaterad till XML-Parser-2.46. Fixes *#4531*.
  - [bdubbs] - Uppdaterad till expat-2.2.9. Fixes *#4532*.
  - [bdubbs] - Uppdaterad till iproute2-5.3.0. Fixes *#4533*.
- 2019-09-24
  - [pierre] - Uppdaterad till linux-5.3.1. Fixes *#4528*.
- 2019-09-14
  - [bdubbs] - Uppdaterad till expat-2.2.8. Fixes *#4527*.
  - [bdubbs] - Uppdaterad till bison-3.4.2. Fixes *#4526*.
  - [bdubbs] - Uppdaterad till linux-5.2.14. Fixes *#4522*.
  - [bdubbs] - Uppdaterad till openssl-1.1.1d. Fixes *#4523*.
  - [bdubbs] - Uppdaterad till sysvinit-2.96. Fixes *#4524*.
  - [bdubbs] - Uppdaterad till tzdata-2019c. Fixes *#4525*.
- 2019-09-02
  - [dj] - Uppdaterad till lfs-bootscripts-20190908.
- 2019-09-02
  - [bdubbs] - Uppdaterad till linux-5.2.11. Fixes *#4517*.
  - [bdubbs] - Uppdaterad till man-db-2.8.7. Fixes *#4518*.
  - [bdubbs] - Uppdaterad till meson-0.51.2. Fixes *#4519*.
  - [bdubbs] - Uppdaterad till findutils-4.7.0. Fixes *#4520*.
  - [dj] - Uppdaterad till LFS-Bootscripts-20190902 - correct LSB dependency information in bootscripts and update standards page for new LSB-Tools package.
- 2019-09-01
  - [bdubbs] - LFS-9.0 released.

1.4. Resurser

1.4.1. FAQ

Om du upptäcker några errors, har några frågor, eller bara upptäcker felstavningar, var vänlig börjar med att utforska denna faq: *http://www.linuxfromscratch.org/faq/*.

1.4.2. Mail-listor

linuxfromscratch.org servern hostar diverse mail-listor vilka används för utvecklandet av LFS-projektet. Dessa listor inkluderar de huvudsakliga listorna för utveckling och användarhjälp. Om du inte finner svar på ditt problem i den faq vilken länkades, är nästa steg att söka igenom mail-listorna: *http://www.linuxfromscratch.org/search.html.*

För information vad gäller de olika listorna, hur man prenumererar, samt diverse annan information, besök: *https://www.linuxfromscratch.org/mail.html*.

1.4.3. IRC

Flertalet av LFS-gemenskapens medlemmar erbjuder assistans via IRC. Innan du söker hjälp via denna kanal, säkerställ vänligen att din fråga inte redan besvarats i faqen eller mail-listorna. Du kan finna IRC-nätverket på *irc.freenode.net*. Kanalens namn är #LFS-support.

1.4.4. Mirror sidor

LFS-projektet använder sig av ett antal mirrors placerade över hela världen, för att öka paketens och hemsidans tillgänglighet. Var vänlig besök LFS hemsida för en lista över nuvarande mirrors.
http://www.linuxfromscratch.org/mirrors.html

1.4.5. Kontakt-information

Var vänlig rikta alla dina frågor samt kommentarer till en av mail-listorna nedan.

1.5. Hjälp

Om du påträffar ett problem medan det att du fortföljer projektet, hantera det enligt nedan process:
#1 – faq. De flesta problem går att finna lösningen på här.
#2 – Om du inte finner lösningen där, försök att felsöka problemet själv.
#3 – Följande fil kan hjälpa: *http://www.linuxfromscratch.org/hints/downloads/files/errors.txt*.
#4 – Ifall ovan inte hjälper, referera till mail-listorna: http://www.linuxfromscratch.org/search.html
#5 – Om inte heller detta hjälper, kontakta supportkanalen som finns på IRC(se sektion 1.4). Alternativt skicka enfråga till en av mail-listorna.

Ifall du överväger att kontakta mail-listorna eller IRC-kanalen, ha då i åtanke att vi får många frågor varje dag och att många av dessa enkelt hade kunnat lösas genom en noggrann läsning av faq eller redan besvarade frågor i mail-listor. Genom att inte skicka in dåligt efterforskade frågor, möjliggör du för oss att fokusera på mer relevanta problem.

1.5.1. Saker att nämna

Bortsett från en kort förklaring vad gäller problemet i sig, är det nödvändigt om du vänder dig till IRC eller en av mail-listorna, att till varje fråga om hjälp bifoga följande information:

- Vilken version av denna bok som används(i detta fall 9.1)
- Värd-distributionen samt versionen som används för att skapa LFS
- Output från Host System Requirements Script
- Paket och/eller sektion som ledde till att problemet uppenbarade sig

- Det exakta error-meddelande eller symptom som inträffar
- Kommentarer vare sig du på något sätt och vis överhuvudtaget har avvikt från instruktionerna i denna bok

Kommentar

Att avvika från denna bok innebär *inte* att vi inte kommer hjälpa dig. LFS handlar om personliga val. Att däremot vara öppen med potentiella avvikelser innebär att vi enklare kan avvgöra vad som är den potentiella anledningen till ditt problem.

1.5.2. Konfigurations-skript (problem)

Om något går fel medan det att du kör **configure** skriptet, undersök då `config.log`filen. Denna kan innehålla info angående problem vilka uppenbarade sig, men vilka inte skrevs till skärmen. Inkludera den *relevanta* infon i din fråga.

1.5.3. Kompilering (problem)

Både skärmoutput samt filens innehåll kan hjälpa vad gäller att finna orsaken till ett kompileringsfel. Skärmoutput från **configure** och **make** skript kan också hjälpa. All information är inte nödvändig att bifoga med din fråga. Nedan återfinns ett exempel på information som är relevant att bifoga.

```
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 detta fall bifogar många endast nedan linje.

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

Detta är inte tillräckligt med information för att framgångsrikt diagnostisera problemet, eftersom det endast noterar att något gick fel, men inte *vad* som gick fel. Hela sektionen borde inkluderas, som i exemplet, eftersom det innehåller kommandot som exekverades samt de relaterade errormeddelandena.

En mycket bra artikel, vad gäller hur man frågor om hjälp vad gäller dessa saker och ting, återfinns här: *http://catb.org/~esr/faqs/smart-questions.*

Del II. Förberedelser för byggnation

Kapitel 2. Att förbereda värdsystemet

2.1. Introduktion

I detta kapitel kommer vi att gå igenom vilka verktyg som krävs för att installera LFS. En partition för att installera LFS på kommer sedan förberedas. Vi kommer skapa denna partition, definiera ett filsystem för den, och sedan montera den.

2.2. Värdsystemets krav

Ditt värdsystem bör ha den följande mjukvaran - med som minimum den version vilken indikeras. Dessa krav borde inte vara några problem för moderna Linuxdistributioner. Notera även att många distributioner placerar headers i separata paket, ofta i form av “<package-name>-devel” eller “<package-name>-dev”. Säkerställ att du installerar dessa ifall den distro du använder tillhandahåller dem.

Tidigare versioner av de paket som listas kan funka, men har inte testats i denna LFS-9.1 kontext.

- **Bash-3.2** (/bin/sh skall vara en symbolisk eller hård link till Bash)
- **Binutils-2.25** (Versioner ovan 2.34 är inte att rekommendera, då de ej har testats)
- **Bison-2.7** (/usr/bin/yacc skall vara en länk till bison eller ett litet skript vilket exekverar bison)
- **Bzip2-1.0.4**
- **Coreutils-6.9**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk skall vara en länk till gawk)
- **GCC-6.2** inkluderat C++ kompilatorn, **g++** (Versioner ovan 9.2.0 är inte att rekommendera, då de ej har testats)

- **Glibc-2.11** (Versioner ovan 2.31 är inte att rekommendera, då de ej har testats)
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Linux Kernel-3.2**

  Anledningen till att kernelversionen är ett krav beror på att vi specifierar denna version, enligt utvecklarnas rekommendation, då vi bygger Glibc i kapitel 6. Det är även ett av udevs krav.

  Om värdsystemets kernelversion är tidigare än 3.2 kommer du behöva ersätta denna kernel med en uppdaterad sådan. Det finns två sätt igenom vilka detta kan göras. Först, se ifall din kernelutgivare tillhandahåller ett paket för version 3.2 eller senare. Ifall inte, kan du kompilera en kernel själv. Instruktioner för att kompilera en kernel samt konfigurera boot-loadern(förutsatt att värden använder GRUB) återfinns i kapitel 8.

- **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**

Viktigt

Notera att symlinksen vilka nämns ovan är en förutsättning för att bygga LFS-systemet på det vis vilket specifieras i denna bok. Symlinks som pekar till annan mjukvara(såsom dash, mawk, etc) kan funka, men är inte testade eller understödda av LFS supportteam, och kan även kräva mer omfattande avvikelser från Instruktionerna, alternativt att fler patchar behöver tillföras till vissa av paketen.

För att säkerställa att ditt system har alla efterfrågade versioner, samt förmågan att kompilera program, kör följande:

```
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

bzip2 --version 2>&1 < /dev/null | head -n1 | cut -d" " -f1,6-
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
ldd --version | head -n1 | cut -d" " -f2-  # glibc version
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. Att bygga LFS i stadier

LFS är designat för att byggas under en session. Detta innebär att systemer utgår ifrån att det inte kommer att stängas av någon gång under själva processen. Detta innebär inte att systemet måste byggas under en session. Problemet är att vissa procedurer måste repeteras efter en omstart.

2.3.1. Kapitel 1–4

Dessa kapitel genomförs på värdsystemet. Då du startar om, var uppmärksam på följande:

- Procedurer genomförda som rootanvändaren efter sektion 2.4 måste ha LFS environment variabeln satt till: *FOR THE ROOT USER*

2.3.2. Kapitel 5

- Partitionen /mnt/lfs måste vara monterad.
- *ALLA* instruktioner i kapitel 5 måste genomföras av användare *lfs*. Ett **su – lfs** ska göras före varje uppgift i kapitel 5.
- Procedurerna i sektion 5.3, “Generella Kompilationsinstruktioner” är kritiska. Ifall det finns något tvivel om hur ett paket bör installeras, säkerställ att alla tidigare extraherade tarballs är borttagna, återextrahera sedan paketfilerna, och slutför instruktionerna i sektionen.

2.3.3. Kapitel 6–8

- Partitionen /mnt/lfs måste vara monterad.
- Vid inträde till chroot, måste LFS environment variabeln vara satt till root. LFS variabeln används inte annars.
- De virtuella filsystemen måste vara monterade. Detta kan göras före eller efter att du äntrat chroot, genom att byta till en av värdens virtuella terminaler och, som root, köra kommandona i sektion 6.2 och 6.2.3.

2.4. Skapandet av en ny partition

Likt de flesta operativsystem är LFS vanligtvis installerat på en dedikerad partition. Det rekommenderade sättet att bygga LFS på är att använda en tillgänglig tom partition eller, ifall du har tillräckligt med utrymme, skapa en ny sådan.

Ett minimalt system kräver en partition med runt 10GB utrymme. Detta är tillräckligt för att lagra alla källkodstarballs samt för att kompilera paketen. Oavsett, om LFS-systemet är tänkt att vara det huvudsakliga Linuxsystemet, måste antagligen extra mjukvara installeras. En 30GB partition är rimligt, för att därmed tillåta ett framtida utökat system.

LFS-systemet i sig självt tar inte upp denna typ av utrymme. En stor del av detta krav handlar om att att tillhandahålla tillräckligt med temporärt utrymme, samt att möjliggöra övrig funktionalitet efter att LFS färdigställts. Utöver detta, så kan mycket utrymme krävas av en kompilerande process. Detta är utrymme som efter slutförd kompilering återlämnas.

Eftersom det inte alltid finns tillräckligt med RAM för kompileringsprocessen, är det en bra idé att använda en mindre diskpartition som en swapdisk. Denna partition används då av kernel för att lagra sällsynt använd data, samt göra mer minne tillgängligt för aktiva processer. Swappartitionen kan vara den samma som för värdsystemet. i sådana fall är det därför inte nödvändigt att skapa en ny.

Starta ett diskpartitionerings program likt **cfdisk** eller **fdisk** med ett kommandolinjeanrop vilket specifierar den hårddisk på vilken den nya partitionen skall skapas – exempelvis */dev/sda* för den primära disken. Skapa en Linux-partition samt, ifall nödvändigt, en swap-partition. Referera till cfdisk(8) och fdisk(8) om du inte vet hur man använder dem.

Kommentar

För erfarna användare finns andra partitioneringsmetoder tillgängliga. LFS-system kan byggas på en mjukvaru RAID-array, eller på en LVM logical volume. Oavsett,så kräver vissa av dessa alternativ en *initramfs*, vilket är ett avancerat ämne. Dessa partitioneringsmetoder är inte rekommenderade för LFS-nybörjare.

Kom ihåg designationen för den nya partitionen, t.ex (`sda5`) Denna bok kommer refera till denna som LFS-partitionen. Kom även ihåg designationen för `swap` partitionen. Dessa designationer kommer behövas för `/etc/fstab` filen.

2.4.1. Andra partitions-problem

Rådfrågning vad gäller partitionering sker ofta på LFS-maillistorna. Detta är ett ytterst subjektivt ämne. Standarden för de flesta distributioner är att använda hela disken, med undantag för en mindre swappartition. Detta är inte optimalt för LFS av ett antal skäl. Det reducerar flexibilitet; gör delandet av data emellan system mer komplicerat; det tar längre tid att göra säkerhetskopior; det kan slösa på systemresurser genom ineffektiv allokering av filsystemets strukturer.

2.4.1.1. Rootpartitionen

En root LFS partition (inte att förväxlas med `/root` mappen) på tio GB är en bra kompromiss för de flesta system. En sådan tillhandahåller tillräckligt med utrymme för att bygga hela LFS och det mesta av BLFS, men är litet nog för att multipla partitioner skall kunna skapas i syfte att experimentera med ett flertal LFS-varianter.

2.4.1.2. Swappartitionen

De flesta distributioner skapar automatiskt en swappartition. Generellt sett är den rekommenderade storleken för denna partition den dubbla mängden RAM som finns tillgänglig, även ifall detta sällan är nödvändigt. Om diskutrymmet är begränsat, sätt då swap-partitionen till 2 GB och monitorera sedan mängden av disk swapping.

Swapping är aldrig bra. Generellt går det avgöra ifall ett system gör det, bara genom att lyssna till diskaktivitet samt observera hur systemet reagerar på kommandon. En första reaktion på potentiell swapping bör vara att kolla efter ett orimligt kommando, ett som exempelvis försöker redigera en 5GB stor fil. Om swapping sker ofta på ditt system, är den rimligaste lösningen att köpa mer RAM.

2.4.1.3. Grub Bios partitionen

Ifall din *boot disk* har partitionerats med en GUID Partition Table (GPT), då måste en liten, vanligtvis 1 MB, partition skapas ifall den inte redan existerar. Denna partition är inte formaterad, men måste vara tillgänglig för användning av GRUB vid installationen av boot loadern. Denna partition kommer vanligtvis benämnas “BIOS Boot” vid användning av **fdisk** eller ha en kod likt *EF02* vid användning av **gdisk**.

Kommentar

Grub Bios partitionen måste återfinnas på disken som BIOS använder för att boota systemet. Detta är inte nödvändigtvis samma diskenhet som var rootpartitionen för LFS är lokaliserad. Diskenheter på ett system kan använda olika varianter av partitionstyper. Kravet för denna partition beror på bootdiskens partitionstyp.

2.4.1.4. Bekvämlighetspartitioner

Det finns flertalet partitioner vilka inte är nödvändiga, men som borde övervägas då ett filsystem designas. Nedan Lista vad gäller detta är inte fullständig, utan tänkt att representera en generell guide.

- /boot – Starkt rekommenderad. Använd denna partition för att lagra kernels och annan bootinfomation. För att minimisera potentiella bootproblem funna hos större diskar, gör då denna partition till den första fysiska sådana på din diskenhet.
- /home – Starkt rekommenderad. Dela din /home och diverse användarkonfigurering över multipla distributioner och LFS-system. Storleken för denna partition är vanligtvis relativt stor, och beror ofta på tillgängligt utrymme.
- /usr – En separat /usr partition används generellt sett vid tillhandahållande av en server för en minimalistisk klient eller ett hårddiskfritt system. Den är normalt sett inte nödvändig för LFS. En storlek på 5GB kan rekommenderas.
- /opt – Denna mapp är huvudsakligen användbar för BLFS, där multipla installationer av större paket likt Gnome och KDE kan installeras utan att sammanfoga filerna med /usr hierarkin. Om den används är 5-10GB rekommenderat.
- /tmp – En separat /tmp mapp är ovanlig, men användbar, då man konfigurerar en minimalistisk klient. Denna partition behöver bara vara ett par GB stor, men är inte nödvändig att definiera.
- /usr/src – Denna partition är väldigt användbar då det kommer till att lagra BLFS-filer och dela dem emellan olika varianter av LFS-system. Det är också en bra plats för att byga BLFS-paket. Bör vara ganska stor, omkring 30-50GB.

Alla separata partitioner vilka du vill ska monteras vid uppstart av systemet, måste specifieras i filen `/etc/fstab`. Detaljer om hur man specifierar partitioner i denna fil, återfinns i sektion 8.2.

2.5. Skapa ett filsystem på partitionen

Då en tom partition nu har skapats, kan filsystemet definieras och skapas. LFS kan använda alla filsystem vilka erkänns av Linux kernel, men de vanligaste varianterna är ext3 och ext4. Valet av filsystem kan vara komplext och beror till stor del på karaktären hos de filer som ska lagras på systemet, samt storleken på partitionen. Till exempel:

ext2
: Passar bra för mindre partitioner såsom /boot.

ext3
: Är en uppgradering till ext2 som inkluderar en journal vilken kan användas för att återställa partitionen ifall en nedstängning går fel. Vanligtvis används ext3 för generella system.

ext4
: Är den senaste versionen inom ext familjen av partitionstyper. Det tillhandahåller flertalet nya aspekter, likt nano-sekunds tidsstämplar, skapande och använding av väldigt stora filer(16TB), samt hastighetsuppgraderingar.

Andra filsystem, likt FAT32, NTFS, ReiserFS, JFS, och XFS är användbara för specialiserade syften. Mer info om dessa filsystem återfinns på: *http://en.wikipedia.org/wiki/Comparison_of_file_systems.*

LFS utgår ifrån att rootfilsystemet (/) är av partitionstypen ext4. För att skapa ett ext4 filsystem på LFS partitionen, kör följande kommando:

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

Om du använder en redan existerande swappartition, finns det ingen anledning att formatera den. Om däremot en ny swappartition skapades, måste den initialiseras med hjälp av nedan kommando:

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

Ersätt *<yyy>* med namnet för swappartitionen.

2.6. Definiera $LFS Variabeln

I denna bok kommer miljövariabeln LFS att användas flertalet gånger. Du bör säkerställa att denna variabel alltid är definierad. Den skall vara satt till namnet på den mapp där du bygger systemet – vi kommer använda /mnt/lfs som exempel, men valet av mapp är upp till dig. Om du bygger LFS på en separat partition, kommer denna mapp vara monteringspunkten för partitionen. Välj en mapp och definiera variabeln med hjälp av nedan kommando:

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

Att ha denna variabel definierad är fördelaktigt då kommandon likt **mkdir -v $LFS/tools** can skrivas. Skalet kommer att ersätta “$LFS” med “/mnt/lfs”(eller vad variabeln än är definierad som), när det processar kommandot.

Varning

Glöm inte, varje gång du lämnat och återvänder till den nuvarande systemmiljön(exempelvis med hjälp av Kommandot “su root”), att säkerställa att $LFS är korrekt definierad. Gör detta med hjälp av kommandot:

```
echo $LFS
```

Säkerställ att output för detta kommando visar pathen till ditt var ditt LFS-system byggs. Om output inte är korrekt, använd dp kommandoexemplet som tidigare på denna sida gavs som exempel. Variabeln $LFS skall då definieras korrekt.

Kommentar

Ett sätt på vilket man kan säkerställa att $LFS alltid är korrekt definierad, är genom att redigera filen “.bash_profile” i både den personliga /home samt i /root/.bash.profile. Applicera till dessa filer export-kommandot ovan. Utöver detta, behöver skalet som specifieras i /etc/passwd filen vara Bash för alla användare vilka behöver använda $LFS. Detta säkerställer att /root/.bash_profile laddas som del av loginprocessen.

En ytterligare sak som man bör ha i åtanke, är metoden som används för att logga in i värdsystemet. Om man loggar in via ett grafiskt gränssnitt, så används vanligtvis inte användarens .bash_profile då en virtuell terminal startas. Lägg i detta fall till exportkommandot till .bashsrc filen för både användare och root. Vissa distros är även instruerade att inte köra .bashsrc instruktioner i en non-interactive bash invocation. Säkerställ att du lägger till exportkommandot innan testet för non-interactive användning.

2.7. Montera den nya partitionen

Då ett filsystem nu har skapats, måste partitionen göras tillgänglig. För att lyckas med detta måste partitionen monteras på en vald monteringspunkt. I denna bok är det förutsatt att filsystemet är monterat under den mapp vilken specifieras vv $LFS miljövariabeln. Såsom beskrivet i den föregående sektionen.

Skapa monteringspunkten och montera partitionen med hjälp av nedan kommandon.

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

Ersätt *<xxx>* med designationen för LFS-partitionen.

Ifall du använder multipla partitioner för LFS – exempelvis en för "/" och en för "/usr" montera dem då enligt nedan:

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

Ersätt *<xxx>* och *<yyy>* med de lämpliga partionsnamnen.

Säkerställ att de nya partitionerna inte är monterade med allt för begränsande rättigheter(som t.ex `nosuid` el. `nodev` alternativen). Kör **mount** kommandot utan några parametrar för att se vilka alternativ som är definierade för de monterade partitionerna. Ifall alternativen nosuid eller nodev är aktiva, måste partitionerna monteras om.

Varning

Instruktionerna ovan utgår ifrån att du inte kommer att starta om din dator under tiden det tar att att slutföra LFS-systemet. Om du stänger av din dator, måste du antingen återmontera partitionerna varje gång du startar upp den igen, eller modifiera värdsystemets /etc/fstab så att montering vid uppstart sker automatiskt. T.ex:

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

Om du använder multipla partitioner, se till att du lägger till så att även dessa monteras.

Om du använder en `swap` partition, säkerställ att den är aktiverad med hjälp av kommandot **swapon**:

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

Ersätt *<zzz>* med namnet på din `swap` partition.

Då det nu finns en etablerad miljö att arbeta inom, är det dags att ladda ned paketen.

Kapitel 3. Paket och patchar

3.1. Introduktion

Detta kapitel inkluderar en lista med paket vilka måste laddas ned för att det ska gå att bygga ett grundläggande system. Versionerna representerar mjukvara vilken vi vet fungerar, och denna bok utgår ifrån användandet av dessa paket. Vi rekommenderar starkt att man undviker att använda nyare versioner, då kommandona för att bygga paketen kanske inte fungerar med nyare versioner. Nyare paket kanske även har diverse problem som måste hanteras. Dessa problem kommer hanteras och lösas i “development”-versionen av boken.

Platserna från vilka det är tänkt att paket ska laddas ned kanske inte alltid är tillgängliga. Ifall de inte längre möjliggör nedladdning, testa ifall du kan använda en sökmotor för att finna paketet. Ifall du inte finner paketet, referera då till: *http://www.linuxfromscratch.org/lfs/packages.html#packages.*

Nedladdade paket och patchar kommer behöva lagras på en plats som under hela byggprocessen är tillgänglig. En arbetsmapp är också nödvändig för att kunna extrahera källkod och bygga programmen. $LFS/sources kan användas både som platsen var tarballs och patchar förvaras, samt some arbetsmapp. Genom att använda denna mapp kommer de nödvändiga paketen att finnas tillgängliga på LFS-partitionen, och därmed alltid vara redo för att användas.

För att skapa denna map, exekvera följande kommando, som användare root, innan det att du påbörjar nedladdningarna.

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

Gör denna mapp skrivbar och klibbig(sticky). “Klibbig” innebär, att även ifall många användare har rättigheter att skriva till mappen, så är det endast ägaren av en fil som har rättigheter för att ta bort denna fil. Följande kommando leder till att denna funktionalitet aktiveras:

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

Ett enkelt sätt att ladda ned alla paket och patchar på, är att använda *wget-list* som input till **wget**. T.ex:

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

Utöver detta finns det en separat fil, *md5sums*, vilken kan köras för att verifiera att alla korrekta paket finns tillgängliga innan det att man fortgår. Placera denna fil i $LFS/sources och kör:

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

3.2. Alla paket

Ladda ned, eller erhåll på annat sätt och vis, följande paket:

• Acl (2.2.53) - 513 KB:
Home page: *https://savannah.nongnu.org/projects/acl*
Download: *http://download.savannah.gnu.org/releases/acl/acl-2.2.53.tar.gz*
MD5 sum: `007aabf1dbb550bcddde52a244cd1070`

• Attr (2.4.48) - 457 KB:
Home page: *https://savannah.nongnu.org/projects/attr*
Download: *http://download.savannah.gnu.org/releases/attr/attr-2.4.48.tar.gz*
MD5 sum: `bc1e5cb5c96d99b24886f1f527d3bb3d`

• Autoconf (2.69) - 1,186 KB:
Home page: *http://www.gnu.org/software/autoconf/*
Download: *http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.xz*
MD5 sum: `50f97f4159805e374639a73e2636f22e`

• Automake (1.16.1) - 1,499 KB:
Home page: *http://www.gnu.org/software/automake/*
Download: *http://ftp.gnu.org/gnu/automake/automake-1.16.1.tar.xz*
MD5 sum: `53f38e7591fa57c3d2cee682be668e5b`

• Bash (5.0) - 9,898 KB:
Home page: *http://www.gnu.org/software/bash/*
Download: *http://ftp.gnu.org/gnu/bash/bash-5.0.tar.gz*
MD5 sum: `2b44b47b905be16f45709648f671820b`

• Bc (2.5.3) - 247 KB:
Home page: *https://github.com/gavinhoward/bc*
Download: *https://github.com/gavinhoward/bc/archive/2.5.3/bc-2.5.3.tar.gz*
MD5 sum: `6582c6fbbae943fbfb8fe14a34feab57`

• Binutils (2.34) - 21,131 KB:
Home page: *http://www.gnu.org/software/binutils/*
Download: *http://ftp.gnu.org/gnu/binutils/binutils-2.34.tar.xz*
MD5 sum: `664ec3a2df7805ed3464639aaae332d6`

• Bison (3.5.2) - 2,308 KB:
Home page: *http://www.gnu.org/software/bison/*
Download: *http://ftp.gnu.org/gnu/bison/bison-3.5.2.tar.xz*
MD5 sum: `49fc2cf23e31e697d5072835e1662a97`

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

• Check (0.14.0) - 753 KB:
Home page: *https://libcheck.github.io/check*
Download: *https://github.com/libcheck/check/releases/download/0.14.0/check-0.14.0.tar.gz*
MD5 sum: `270e82a445be6026040267a5e11cc94b`

• Coreutils (8.31) - 5,284 KB:
Home page: *http://www.gnu.org/software/coreutils/*
Download: *http://ftp.gnu.org/gnu/coreutils/coreutils-8.31.tar.xz*
MD5 sum: `0009a224d8e288e8ec406ef0161f9293`

• DejaGNU (1.6.2) - 514 KB:
Home page: *http://www.gnu.org/software/dejagnu/*
Download: *http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.6.2.tar.gz*
MD5 sum: `e1b07516533f351b3aba3423fafeffd6`

• Diffutils (3.7) - 1,415 KB:
Home page: *http://www.gnu.org/software/diffutils/*
Download: *http://ftp.gnu.org/gnu/diffutils/diffutils-3.7.tar.xz*
MD5 sum: `4824adc0e95dbbf11dfbdfaad6a1e461`

• E2fsprogs (1.45.5) - 7,753 KB:
Home page: *http://e2fsprogs.sourceforge.net/*
Download: *https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.45.5/e2fsprogs-1.45.5.tar.gz*
MD5 sum: `6d35428e4ce960cb7e875afe5849c0f3`

• Elfutils (0.178) - 8,797 KB:
Home page: *https://sourceware.org/ftp/elfutils/*
Download: *https://sourceware.org/ftp/elfutils/0.178/elfutils-0.178.tar.bz2*
MD5 sum: `5480d0b7174446aba13a6adde107287f`

• Eudev (3.2.9) - 1,914 KB:
Download: *https://dev.gentoo.org/~blueness/eudev/eudev-3.2.9.tar.gz*
MD5 sum: `dedfb1964f6098fe9320de827957331f`

• Expat (2.2.9) - 413 KB:
Home page: *https://libexpat.github.io/*
Download: *https://prdownloads.sourceforge.net/expat/expat-2.2.9.tar.xz*
MD5 sum: `d2384fa607223447e713e1b9bd272376`

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

• File (5.38) - 911 KB:
Home page: *https://www.darwinsys.com/file/*
Download: *ftp://ftp.astron.com/pub/file/file-5.38.tar.gz*
MD5 sum: `3217633ed09c7cd35ed8d04191675574`

Kommentar

Fil (5.38) kanske inte längre är tillgänglig på denna plats. Sitens administratör tar då och då bort äldre versioner när nya släpps. Det är möjligt att du kan hitta den korrekt versionen via: http://www.linuxfromscratch.*org/lfs/download.html#ftp.*

• Findutils (4.7.0) - 1,851 KB:
Home page: *http://www.gnu.org/software/findutils/*
Download: *http://ftp.gnu.org/gnu/findutils/findutils-4.7.0.tar.xz*
MD5 sum: `731356dec4b1109b812fecfddfead6b2`

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

• Gawk (5.0.1) - 3,063 KB:
Home page: *http://www.gnu.org/software/gawk/*
Download: *http://ftp.gnu.org/gnu/gawk/gawk-5.0.1.tar.xz*
MD5 sum: `f9db3f6715207c6f13719713abc9c707`

• GCC (9.2.0) - 68,953 KB:
Home page: *https://gcc.gnu.org/*
Download: *http://ftp.gnu.org/gnu/gcc/gcc-9.2.0/gcc-9.2.0.tar.xz*
MD5 sum: `3818ad8600447f05349098232c2ddc78`

• GDBM (1.18.1) - 920 KB:
Home page: *http://www.gnu.org/software/gdbm/*
Download: *http://ftp.gnu.org/gnu/gdbm/gdbm-1.18.1.tar.gz*
MD5 sum: `988dc82182121c7570e0cb8b4fcd5415`

• Gettext (0.20.1) - 9,128 KB:
Home page: *http://www.gnu.org/software/gettext/*
Download: *http://ftp.gnu.org/gnu/gettext/gettext-0.20.1.tar.xz*
MD5 sum: `9ed9e26ab613b668e0026222a9c23639`

• Glibc (2.31) - 16,286 KB:
Home page: *http://www.gnu.org/software/libc/*
Download: *http://ftp.gnu.org/gnu/glibc/glibc-2.31.tar.xz*
MD5 sum: `78a720f17412f3c3282be5a6f3363ec6`

• GMP (6.2.0) - 1,966 KB:
Home page: *http://www.gnu.org/software/gmp/*
Download: *http://ftp.gnu.org/gnu/gmp/gmp-6.2.0.tar.xz*
MD5 sum: `a325e3f09e6d91e62101e59f9bda3ec1`

• Gperf (3.1) - 1,188 KB:
Home page: *http://www.gnu.org/software/gperf/*
Download: *http://ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz*
MD5 sum: `9e251c0a618ad0824b51117d5d9db87e`

• Grep (3.4) - 1,520 KB:
Home page: *http://www.gnu.org/software/grep/*
Download: *http://ftp.gnu.org/gnu/grep/grep-3.4.tar.xz*
MD5 sum: `111b117d22d6a7d049d6ae7505e9c4d2`

• Groff (1.22.4) - 4,044 KB:
Home page: *http://www.gnu.org/software/groff/*
Download: *http://ftp.gnu.org/gnu/groff/groff-1.22.4.tar.gz*
MD5 sum: `08fb04335e2f5e73f23ea4c3adbf0c5f`

• GRUB (2.04) - 6,245 KB:
Home page: *http://www.gnu.org/software/grub/*
Download: *https://ftp.gnu.org/gnu/grub/grub-2.04.tar.xz*
MD5 sum: `5aaca6713b47ca2456d8324a58755ac7`

• Gzip (1.10) - 757 KB:
Home page: *http://www.gnu.org/software/gzip/*
Download: *http://ftp.gnu.org/gnu/gzip/gzip-1.10.tar.xz*
MD5 sum: `691b1221694c3394f1c537df4eee39d3`

• Iana-Etc (2.30) - 201 KB:
Home page: *http://freecode.com/projects/iana-etc*
Download: *http://anduin.linuxfromscratch.org/LFS/iana-etc-2.30.tar.bz2*
MD5 sum: `3ba3afb1d1b261383d247f46cb135ee8`

• Inetutils (1.9.4) - 1,333 KB:
Home page: *http://www.gnu.org/software/inetutils/*
Download: *http://ftp.gnu.org/gnu/inetutils/inetutils-1.9.4.tar.xz*
MD5 sum: `87fef1fa3f603aef11c41dcc097af75e`

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

• IPRoute2 (5.5.0) - 731 KB:
Home page: *https://www.kernel.org/pub/linux/utils/net/iproute2/*
Download: *https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-5.5.0.tar.xz*
MD5 sum: `ee8e2cdb416d4a8ef39525d39ab7c2d0`

• Kbd (2.2.0) - 1,090 KB:
Home page: *http://ftp.altlinux.org/pub/people/legion/kbd*
Download: *https://www.kernel.org/pub/linux/utils/kbd/kbd-2.2.0.tar.xz*
MD5 sum: `d1d7ae0b5fb875dc082731e09cd0c8bc`

• Kmod (26) - 540 KB:
Download: *https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-26.tar.xz*
MD5 sum: `1129c243199bdd7db01b55a61aa19601`

• Less (551) - 339 KB:
Home page: *http://www.greenwoodsoftware.com/less/*
Download: *http://www.greenwoodsoftware.com/less/less-551.tar.gz*
MD5 sum: `4ad4408b06d7a6626a055cb453f36819`

• LFS-Bootscripts (20191031) - 32 KB:
Download: *http://www.linuxfromscratch.org/lfs/downloads/9.1/lfs-bootscripts-20191031.tar.xz*
MD5 sum: `e9249541960df505e4dfac0c32369372`

• Libcap (2.31) - 97 KB:
Home page: *https://sites.google.com/site/fullycapable/*
Download: *https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.31.tar.xz*
MD5 sum: `52120c05dc797b01f5a7ae70f4335e96`

• Libffi (3.3) - 1,275 KB:
Home page: *https://sourceware.org/libffi/*
Download: *ftp://sourceware.org/pub/libffi/libffi-3.3.tar.gz*
MD5 sum: `6313289e32f1d38a9df4770b014a2ca7`

• Libpipeline (1.5.2) - 971 KB:
Home page: *http://libpipeline.nongnu.org/*
Download: *http://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.2.tar.gz*
MD5 sum: `169de4cc1f6f7f7d430a5bed858b2fd3`

• Libtool (2.4.6) - 951 KB:
Home page: *http://www.gnu.org/software/libtool/*
Download: *http://ftp.gnu.org/gnu/libtool/libtool-2.4.6.tar.xz*
MD5 sum: `1bfb9b923f2c1339b4d2ce1807064aa5`

• Linux (5.5.3) - 108,112 KB:
Home page: *https://www.kernel.org/*
Download: *https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.5.3.tar.xz*
MD5 sum: `3ea50025d8c679a327cf2fc225d81a46`

Kommentar

Linux kernel uppdateras relativt ofta. Det är frekvent förekommande att detta beror på upptäckta säkerhetshål. Den senast tillgängliga 5.5.x kerneln bör användas, ifall inte annat nämns i Erratan.

• M4 (1.4.18) - 1,180 KB:
Home page: *http://www.gnu.org/software/m4/*
Download: *http://ftp.gnu.org/gnu/m4/m4-1.4.18.tar.xz*
MD5 sum: `730bb15d96fffe47e148d1e09235af82`

• Make (4.3) - 2,263 KB:
Home page: *http://www.gnu.org/software/make/*
Download: *http://ftp.gnu.org/gnu/make/make-4.3.tar.gz*
MD5 sum: `fc7a67ea86ace13195b0bce683fd4469`

• Man-DB (2.9.0) - 1,814 KB:
Home page: *https://www.nongnu.org/man-db/*
Download: *http://download.savannah.gnu.org/releases/man-db/man-db-2.9.0.tar.xz*
MD5 sum: `897576a19ecbef376a916485608cd790`

• Man-pages (5.05) - 1,649 KB:
Home page: *https://www.kernel.org/doc/man-pages/*
Download: *https://www.kernel.org/pub/linux/docs/man-pages/man-pages-5.05.tar.xz*
MD5 sum: `da25a4f8dfed0a34453c90153b98752d`

• Meson (0.53.1) - 1,516 KB:
Home page: *https://mesonbuild.com*
Download: *https://github.com/mesonbuild/meson/releases/download/0.53.1/meson-0.53.1.tar.gz*
MD5 sum: `9bf73f7b5a2426a7c8674a809bb8cae2`

• MPC (1.1.0) - 685 KB:
Home page: *http://www.multiprecision.org/*
Download: *https://ftp.gnu.org/gnu/mpc/mpc-1.1.0.tar.gz*
MD5 sum: `4125404e41e482ec68282a2e687f6c73`

• MPFR (4.0.2) - 1,409 KB:
Home page: *https://www.mpfr.org/*
Download: *http://www.mpfr.org/mpfr-4.0.2/mpfr-4.0.2.tar.xz*
MD5 sum: `320fbc4463d4c8cb1e566929d8adc4f8`

• Ninja (1.10.0) - 206 KB:
Home page: *https://ninja-build.org/*
Download: *https://github.com/ninja-build/ninja/archive/v1.10.0/ninja-1.10.0.tar.gz*
MD5 sum: `cf1d964113a171da42a8940e7607e71a`

• Ncurses (6.2) - 3,346 KB:
Home page: *http://www.gnu.org/software/ncurses/*
Download: *http://ftp.gnu.org/gnu/ncurses/ncurses-6.2.tar.gz*
MD5 sum: `e812da327b1c2214ac1aed440ea3ae8d`

• OpenSSL (1.1.1d) - 8,639 KB:
Home page: *https://www.openssl.org/*
Download: *https://www.openssl.org/source/openssl-1.1.1d.tar.gz*
MD5 sum: `3be209000dbc7e1b95bcdf47980a3baa`

• Patch (2.7.6) - 766 KB:
Home page: *https://savannah.gnu.org/projects/patch/*
Download: *http://ftp.gnu.org/gnu/patch/patch-2.7.6.tar.xz*
MD5 sum: `78ad9937e4caadcba1526ef1853730d5`

• Perl (5.30.1) - 12,078 KB:
Home page: *https://www.perl.org/*
Download: *https://www.cpan.org/src/5.0/perl-5.30.1.tar.xz*
MD5 sum: `f399f3aaee90ddcff5eadd3bccdaacc0`

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

• Procps (3.3.15) - 884 KB:
Home page: *https://sourceforge.net/projects/procps-ng*
Download: *https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-3.3.15.tar.xz*
MD5 sum: `2b0717a7cb474b3d6dfdeedfbad2eccc`

• Psmisc (23.2) - 297 KB:
Home page: *http://psmisc.sourceforge.net/*
Download: *https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.2.tar.xz*
MD5 sum: `0524258861f00be1a02d27d39d8e5e62`

• Python (3.8.1) - 17,411 KB:
Home page: *https://www.python.org/*
Download: *https://www.python.org/ftp/python/3.8.1/Python-3.8.1.tar.xz*
MD5 sum: `b3fb85fd479c0bf950c626ef80cacb57`

• Python Documentation (3.8.1) - 6,374 KB:
Download: *https://www.python.org/ftp/python/doc/3.8.1/python-3.8.1-docs-html.tar.bz2*
MD5 sum: `edc8c97f9680373fcc1dd952f0ea7fcc`

• Readline (8.0) - 2,907 KB:
Home page: *https://tiswww.case.edu/php/chet/readline/rltop.html*
Download: *http://ftp.gnu.org/gnu/readline/readline-8.0.tar.gz*
MD5 sum: `7e6c1f16aee3244a69aba6e438295ca3`

• Sed (4.8) - 1,317 KB:
Home page: *http://www.gnu.org/software/sed/*
Download: *http://ftp.gnu.org/gnu/sed/sed-4.8.tar.xz*
MD5 sum: `6d906edfdb3202304059233f51f9a71d`

• Shadow (4.8.1) - 1,574 KB:
Download: *https://github.com/shadow-maint/shadow/releases/download/4.8.1/shadow-4.8.1.tar.xz*
MD5 sum: `4b05eff8a427cf50e615bda324b5bc45`

• Sysklogd (1.5.1) - 88 KB:
Home page: *http://www.infodrom.org/projects/sysklogd/*
Download: *http://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.1.tar.gz*
MD5 sum: `c70599ab0d037fde724f7210c2c8d7f8`

• Sysvinit (2.96) - 120 KB:
Home page: *https://savannah.nongnu.org/projects/sysvinit*
Download: *http://download.savannah.gnu.org/releases/sysvinit/sysvinit-2.96.tar.xz*
MD5 sum: `48cebffebf2a96ab09bec14bf9976016`

• Tar (1.32) - 2,055 KB:
Home page: *http://www.gnu.org/software/tar/*
Download: *http://ftp.gnu.org/gnu/tar/tar-1.32.tar.xz*
MD5 sum: `83e38700a80a26e30b2df054e69956e5`

• Tcl (8.6.10) - 9,907 KB:
Home page: *http://tcl.sourceforge.net/*
Download: *https://downloads.sourceforge.net/tcl/tcl8.6.10-src.tar.gz*
MD5 sum: `97c55573f8520bcab74e21bfd8d0aadc`

• Texinfo (6.7) - 4,237 KB:
Home page: *http://www.gnu.org/software/texinfo/*
Download: *http://ftp.gnu.org/gnu/texinfo/texinfo-6.7.tar.xz*
MD5 sum: `d4c5d8cc84438c5993ec5163a59522a6`

• Time Zone Data (2019c) - 383 KB:
Home page: *https://www.iana.org/time-zones*
Download: *https://www.iana.org/time-zones/repository/releases/tzdata2019c.tar.gz*
MD5 sum: `f6987e6dfdb2eb83a1b5076a50b80894`

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

• Util-linux (2.35.1) - 5,018 KB:
Home page: *http://freecode.com/projects/util-linux*
Download: *https://www.kernel.org/pub/linux/utils/util-linux/v2.35/util-linux-2.35.1.tar.xz*
MD5 sum: `7f64882f631225f0295ca05080cee1bf`

• Vim (8.2.0190) - 14,406 KB:
Home page: *https://www.vim.org*
Download: *http://anduin.linuxfromscratch.org/LFS/vim-8.2.0190.tar.gz*
MD5 sum: `f5337b1170df90e644a636539a0313a3`

Kommentar
Vims version ändras dagligen. För den senaste versionen, besök: *https://github.com/vim/vim/releases*.

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

• Xz Utils (5.2.4) - 1030 KB:
Home page: *https://tukaani.org/xz*
Download: *https://tukaani.org/xz/xz-5.2.4.tar.xz*
MD5 sum: `003e4d0b1b1899fc6e3000b24feddf7c`

• Zlib (1.2.11) - 457 KB:
Home page: *https://www.zlib.net/*
Download: *https://zlib.net/zlib-1.2.11.tar.xz*
MD5 sum: `85adef240c5f370b308da8c938951a68`

• Zstd (1.4.4) - 1,903 KB:
Home page: *https://facebook.github.io/zstd/*
Download: *https://github.com/facebook/zstd/releases/download/v1.4.4/zstd-1.4.4.tar.gz*
MD5 sum: `487f7ee1562dee7c1c8adf85e2a63df9`

Total storlek för dessa paket: omkring 398 MB

3.3. Nödvändiga patchar

Utöver paketen, är det även nödvändigt att ladda ned ett antal patchar. Dessa patchar korrigerar misstag i paketen, vilka bör fixas av de som underhåller dem. Dessa patchar gör även mindre förändringar i syfte att underlätta hanteringen av paketen. Följande patchar behövs för att bygga ett LFS-system:

• Bash Upstream Fixes Patch - 22 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/9.1/bash-5.0-upstream_fixes-1.patch*
MD5 sum: `c1545da2ad7d78574b52c465ec077ed9`

• Bzip2 Documentation Patch - 1.6 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/9.1/bzip2-1.0.8-install_docs-1.patch*
MD5 sum: `6a5ac7e89b791aae556de0f745916f7f`

• Coreutils Internationalization Fixes Patch - 168 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/9.1/coreutils-8.31-i18n-1.patch*
MD5 sum: `a9404fb575dfd5514f3c8f4120f9ca7d`

• Glibc FHS Patch - 2.8 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/9.1/glibc-2.31-fhs-1.patch*
MD5 sum: `9a5997c3452909b1769918c759eff8a2`

• Kbd Backspace/Delete Fix Patch - 12 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/9.1/kbd-2.2.0-backspace-1.patch*
MD5 sum: `f75cca16a38da6caa7d52151f7136895`

• Sysvinit Consolidated Patch - 2.4 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/9.1/sysvinit-2.96-consolidated-1.patch*
MD5 sum: `4900322141d493e74020c9cf437b2cdc`

Total storlek för dessa patchar: omkring 208.8 KB

Utöver de ovan nämnda patcharna, existerar det ett antal valbara patchar skapade av LFS-gemenskapen. Dessa valbara patchar löser mindre problem eller tillför funktionalitet vilken inte är tillgänglig som standard. Känn dig fri att utforska databasen för patchar och ladda ned de vilka du anser kan vara av nytta för ditt system:
http://www.linuxfromscratch.org/patches/downloads/

Kapitel 4. Slutgiltiga förberedelser

4.1. Introduktion

I detta kapitel kommer vi utföra ett antal uppgifter, som förberedelse för att bygga det temporära systemet. Vi kommer att skapa en mapp i $LFS för installationen av de temporärar verktygen, samt en opriviligerad användare i syfte att minimerar risk, samt skapa en lämplig byggmiljö för denna användare. Vi kommer även att definiera det tidsmått(SBU) vilket kommer användas som måttstock för hur lång tid paket tar att bygga. Även info om testsviterna kommer att ges.

4.2. Skapa $LFS/tools mappen

Alla program kompilerade i kapitel 5 kommer att installeras i `$LFS/tools` i syfter att lagra dem separat från program kompilerade i kapitel 6. De program vilka kompileras här är temorära verktyg, och inte del av det slutgiltiga systemet. Genom att lagra dessa program i en separat mapp, är det även enkelt att ta bort dem efter användning. Det förhindrar även att de av misstag hamnar i värdens produktionsmappar(kan enkelt ske av misstag i kapitel 5).

Skapa den efterfrågade mappen genom att som root köra:

```
mkdir -v $LFS/tools
```

Nästa steg är att skapa en `/tools` symlink på värdsystemet. Denna kommer peka till den nyligen skapade mappen på LFS-partitionen. Kör även detta kommando som root:

```
ln -sv $LFS/tools /
```

Kommentar

Ovan kommando är korrekt. **ln** kommandot har ett antal syntaktiska variationer, så utforska **info coreutils ln** samt ln(1) innan du rapporterar in ett error.

Den skapade symlinken möjliggör för toolchainen att kompileras på ett sådant vis att den alltid pekar till */tools*, vilket innebär att kompilator, assembler, samt länkare kommer fungera både i kapitel 5(då vi ännu använder ett antal verktyg från värden) och i kapitel 6(när vi är “chrooted” i LFS-partitionen).

4.3. Skapa LFS användaren

Då man är inloggad som root, kan ett enda misstag leda till att hela systemet skadas eller förstörs. Vi rekommenderar därför att paketen i nästa kapitel byggs av en opriviligerad användare. Du kan använda din egen användare, men för att göra det enklare att etablera en ren arbetsmiljö, skapa en användare vid namn(“lfs”) som medlem av en ny grupp(“lfs”). Använd denna användare under installationsprocessen. Som root, kör följande kommando:

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

Vad alternativen till kommandot betyder:

`-s /bin/bash`
Gör **bash** till standardskal för användare ‘lfs’ .

`-g lfs`
Detta alternativ lägger till användare ‘lfs’ till grupp ‘lfs’

-m
Detta skapar en hemkatalog för användaren.

-k /dev/null
Denna parameter förhindrar potentiell kopiering av filer från en skelettmapp(standard är /etc/skel) genom att ändra Input-platsen till den särskilda null-enheten.

lfs
Själva namnet på användaren samt på gruppen.

För att logga in som lfs (i motsats till att byta till användare lfs då loggad in som root, vilket inte kräver att användaren har ett lösenord), specifiera ett lösenord för lfs.

```
passwd lfs
```

Ge lfs full tillgång till $LFS/tools genom att gör lfs till mappens ägare.

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

Om, som rekommenderat, en separat arbetsmapp skapades, ge användaren lfs ägarskap över denna mapp.

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

Kommentar

I vissa värdsystem kommer följande kommando inte att slutföras korrekt, och förpassar inloggning av lfs-användaren till bakgrunden. Ifall skalets användare inte skiftar till “lfs:~$” omgående, kan detta lösas med hjälp av kommandot **fg**.

Logga nu in som användaren lfs. Detta kan göras via en virtuell konsol, genom en display manager, alternativt med följande kommando:

```
su - lfs
```

Symbolen “-” instruerar **su** att starta ett login-skal, i motsats till ett icke-login-skal. Skilladerna emellan dessa två typer av skal kan efterforskas i bash(1) samt **info bash**.

4.4. Etablerandet av miljön

Etablera en en god arbetsmiljö genom att skapa två nya uppstartsskript för **bash** skalet. Medan loggad in som lfs utför följande kommando för att skapa en ny .bash_profile:

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

Då loggad in som användare lfs, är det initiala skalet vanligtvis ett login-skal vilket läser värdens /etc /profile (vilken antagligen innehåller inställningar och miljövariabler) och sedan .bash_profile. Kommandot **exec env -i.../bin/bash** i .bash_profile ersätter det körda skalet med ett nytt sådant som har en komplett tom miljö(bortsett från HOME, TERM och PS1 variablerna). Detta säkerställer att inga oönskade och potentiellt farliga miljö-variabler läcker in, från värden, till byggmiljön. Tekniken som används här säkerställer en ren miljö.

Denna nya instans av skalet är ett icke-login-skal, vilket inte läser från filer`/etc/profile` och `.bash_profile`. Istället läser detta skal från filen .bashrc. Skapa därför filen .bashrc:

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
LFS_TGT=$(uname -m)-lfs-linux-gnu
PATH=/tools/bin:/bin:/usr/bin
export LFS LC_ALL LFS_TGT PATH
EOF
```

Kommandot **set +h** stänger av **bash**s hash-funktion. Hashing är vanligtvis en användbar funktion – **bash** använder ett hashtable för att minnas de kompletta sökvägarna till exekuterbara filer, för att undvika att söka efter dessa sökvägar om och om igen. I vilket fall, de nya verktygen bör användas så snart de installerats. Genom att stänga av funktionen för hash, kommer skalet alltid söka igenom PATH när ett program skall köras. Till följd av detta kommer skalet hitta de nya verktygen så snart som de finns tillgängliga, istället för att använda sökvägen till en tidigare sparad version.

Att sätta user file-creation mask (umask) till 022 säkerställer att skapade filer och mappar endast är skrivbara till av dess ägare, men är läsbara och exekuterbara av alla (förutsatt att standard inställningar används av open(2) systemanropet, kommer nya filer att tillskrivas 644 och nya mappar 755).

$LFS-variabeln bör sättas till den valda monteringspunkten.

LC_ALL-variabeln kontrollerar lokaliseringen av vissa program, för att säkerställa att deras meddelanden följer de konventioner som existerar för diverse länder. Att sätta LC_ALL till “POSIX” eller “C” (de två är likadana), innebär att allt i chrootmiljön kommer att fungera som förväntat.

LFS_TGT-variabeln sätter en ickestandard, men kompatibel maskinbeskrivning för använding av vår crosscompiler samt länkare och då vi crosskompilerar vår temporära toolchain. Mer information om detta återfinns under sektion 5.2

Genom att placera /tools /bin för standard PATH, kommer alla paket som installeras i kapitel 5 att identifieras av skalet omedelbart efter det att de installerats. Detta, kombinerat med att bashs hashfunktion har deaktiverats, begränsar risken för att gamla program används (från värden) när samma program redan finns tillgängliga i kapitel 5s miljö.

Slutligen, för att fullt ut förbereda för byggandet av de temporära verktygen, kör följande kommando:

```
source ~/.bash_profile
```

4.5. Angående SBUs

Det är många som på förhand vill veta hur lång tid det ungefär kommer ta att kompilera samt installera varje paket. Eftersom Linux From Scratch kan byggas på många olika system, är det omöjligt att tillhandahålla korrekta angivelser. Det största paketet (glibc) tar ca 20 minuter att bygga på de snabbaste systemen, men kan ta upp till 3 dagar på system vilka är långsammare. Vi kommer därför att använda ett mått kallat Standard Build Unit(SBU).

Måttet SBU fungerar enligt följande. Det första paket som kompileras i denna bok är binutils i kapitel 5. Den tid vilken det tar att kompilera detta paket är vad som kommer att refereras till som en SBU. All annan kompileringstid kommer att placeras i denna kontext.

Föreställ dig exempelvis ett paket med en byggtid på 4.5 SBUs. Ifall det tog 10 minuter att kompilera och installera den första rundan av binutils, så kommer det ta *ungefär* 45 minuter att bygga och installera detta exempelpaket. Lyckligtvis är de flesta pakets byggtider kortare än den för binutils.

Generellt sett är SBUs inte helt representativa eftersom de beror på så många omkringliggande faktorer, däribland inkluderat värdsystemets version av GCC. De ger en relativ uppfattning om hur lång tid det kan tänkas ta att bygga ett paket, men skillnaden i tid kan variera kraftigt.

Kommentar

För många moderna system med multipla processorer (eller kärnor) kan kompileringstiden för ett paket reduceras genom att man genomför en “parallel make” genom att antingen sätta en miljövariabel eller genom att instruera **make** hur många kärnor som finns tillgängliga. Exempelvis kan en Core2Duo understödja 2 parallella processer, ifall nedan variabeldefinition exporteras:

```
export MAKEFLAGS='-j 2'
```

Eller så kan man bara bygga med flaggan:

```
make -j2
```

När multipla processorer används på detta vis, kommer SBUs vilka nämns i boken skifta på ett kraftigare sätt än de vanligtvis skulle. Och i vissa fall kommer make helt enkelt misslyckas. Att analysera outputen för make blir även svårare, då meddelanden från processerna kommer att blandas ihop. Om du använder multipla kärnor och får problem, gör om steget med endast en processor som bygger.

4.6. Angående Testsviterna

De flesta paket tillhandahåller en testsvit. Att köra denna svit för ett paket som just byggts är ofta en bra idé, då det innebär ett sorts kontroll vilken bekräftar att allt kompilerades korrekt. En testsvit som returnerar förväntade värden bevisar vanligtvis att ett paket fungerar som utvecklarna har tänkt sig. Däremot garanterar det inte att paketet är helt och hållet fritt från buggar.

Vissa testsviter är viktigare än andra. Testsviterna för de grundläggande paketen för vår toolchain – GCC, Binutils, Glibc – är exempelvis väldigt viktiga, till följd av dessa pakets centrala funktioner i ett Linuxsystem. Sviterna för GCC och Glibc kan ta en väldigt lång tid att slutföra, men det är starkt rekommenderat att dessa ändå körs.

Kommentar

Enligt erfarenhet finns det inte mycket att tjäna på att köra testsviterna i kapitel 5. Detta beror på att värdsystemet kan har en kraftfull påverkan på hur testen körs, något som ofta leder till nästan oförklarliga rapporterade misslyckanden. Eftersom verktygen i kapitel 5 endast är temporära, rekommenderar vi inte att den genomsnittlige läsaren kör dessa testsviter. Instruktioner för att köra dem medföljer, men det är helt och hållet frivilligt vare sig man gör så eller inte.

Ett vanligt problem då man kör testsviterna för Binutils och GCC, är att man får slut på pseudoterminaler(PTYs). detta kan resultera i ett stort antal misslyckade test, och kan ske på grund av ett antal orsaker. Den mest sannolika sådana är att värd-systemet inte har devpts-filsystemet korrekt implementerat. Detta är en fråga vilken mer detaljerat diskuteras här: *www.linuxfromscratch.org/lfs/faq.html#no-ptys.*

Ibland kommer testsviter misslyckas, men på grund av orsaker som utvecklarna är medvetna om och har bedömt som icke kritiska. Konsultera loggen nedan, för att verifiera vare sig dessa misslyckande är förväntade eller inte. Denna sida Är giltig för alla test-sviter i denna bok.
http://www.linuxfromscratch.org/lfs/build-logs/9.1/

Kapitel 5. Konstruerandet av ett temporärt system

5.1. Introduktion

Detta kapitel visar hur man bygger ett minimalt Linuxsystem. Detta system kommer innehålla precis den mängd av verktyg vilka krävs för att man ska kunna börja konstruera det slutgiltiga systemet i kapitel 6, samtidigt som det utökar användarvänligheten bortom den som en verkligt minimal miljö skulle tillhandahålla.

Att bygga detta minimala system innebär två steg. Det första steget är att bygga en ny och hostindependent toolchain. Detta inkluderar kompilator, assembler, länkare, bibiliotek, samt diverse verktyg. i nästa steg används denna toolchain för att bygga de övriga nödvändiga verktygen.

Filerna vilka kompileras i detta kapitel kommer att installeras under mappen $LFS/tools, så att de lagras separat från de filer vilka installeras i kapitel 6. Eftersom paketen som kompileras här är temporära, vill vi inte lagra dem permanent.

5.2. Toolchain tekniska kommentarer

Denna sektion beskriver den logiska grunden bakom byggmetoderna, samt diskuterar diverse tekniska detaljer. Det är inte nödvändigt att direkt förstå allt i denna sektion. Informationen här kommer vara enklare att ta till sig efter det att en byggnation av LFS redan genomförts. Denna sektion kan däremot refereras till under själva processen.

Det övergripandet målet i kapitel 5 är att producera en temporär miljö vilken tillgängliggör ett antal verktyg vilka kan isoleras från värdsystemet. Genom att använda **chroot** kommer kommandona i de efterföljande kapitlen att isoleras inom den miljön, vilket säkerställer en ren och problemfri byggnation av LFS-systemet. Processen i sig har designats för att minimerar riskerna för nya läsare, samtidigt som den ska leda till så mycket tillgodogjord kunskap som möjligt.

Kommentar

Innan det att du fortsätter, var medveten om namnet på den plattform du arbetar ifrån, ofta hänvisad till som “target triplet”. Ett enkelt sätt att identifiera detta namn är att köra skriptet **config.guess,** vilket inkluderas i många paket. Extrahera Binutils sources och kör skriptet: **./config.guess** och notera outputen.

Ha även kunskap om plattformens dynamiska länkare, ofta refererad till som dynamisk laddare (inte att Blandas ihop med standardlänkaren **ld** vilken är del av Binutils). Den dynamiska länkaren tillhandahållen av Glibc hittar och laddar de delade biblioteken vilka ett program behöver, förbereder programmet att köras, och kör det sedan. Namnet för ett 32bit-system kommer vara ld-linux.so.2 (ld-linux-x86-64.so.2 för 64-bit).
Ett definitivt sätt att kontrollera namnet för den dynamiska länkaren på, är att inspektera en slumpmässig binär fil genom att köra: “readelf -l <name of binary> | grep interpreter” och notera outputen.
Den källa som överblickar alla plattformar återfinns i filen shlib-versions i Glibc source tree.

Ett antal tekniska kommentarer vad gäller kapitel 5s byggmetoder:

- Att justera namnet på den arbetande plattformen, genom att ändra “vendor”-fältets target triplet med hjälp av LFS_TGT-variabeln, säkerställer att den första byggnationen av Binutils och GCC producerar en kompatibel cross-länkare och cross-kompilator. Istället för att producera binärer för andra typer utav arkitektur, kommer cross-länkaren och cross-kompilatorn att producera binärer kompatibla med den aktuella hårdvaran.

- De temporärara biblioteken är cross-kompilerade. Eftersom en cross-kompilator inte kan förlita sig på något från dess värdsystem, innebär denna metod att man utesluter potentiell kontaminering av byggsystemet. Detta sker genom att man minimerar sannolikheten för att headers och bibliotek från värden inkorporeras i de nya verktygen. Denna typ av kompilering möjliggör även byggandet av både 32bit och 64bit bibliotek, på ett 64bit system.
- Försiktig manipulering av GCC source meddelar kompilatorn vilken dynamisk länkare som kommer användas.

Binutils installeras först eftersom **configure** körningarna av både GCC och Glibc genomgör diverse test på assemblern och länkaren, för att avgöra vilka mjukvarufunktioner att aktivera och avaktivera. Detta är viktigare än man kanske först tänker sig. En GCC eller Glibc som konfigurerats på ett inkorrekt vis kan resultera i en dolt trasig toolchain, där effekterna av sådana problem kanske inte visar sig förräns det att systemet nästan är färdigt. En körning av testsviterna kommer vanligtvis, innan allt för mycket jobb krävs, uppmärksamma användaren på att något är fel.

Binutils installerar sin assembler inuti två separata platser: /tools/bin och /tools/$LFS_TGT/bin Verktygen i en plats är hård-länkade till den andra. En viktig aspekt av länkaren är dess library search order. Detaljerad information kan erhållas från **ld** genom att använda flaggan --verbose. Exempelvis **ld --verbose | grep SEARCH** kommer illustrera de nuvarande sökvägarna samt den ordning vilka de används i. Den visar vilka filer som är länkade med **ld** genom att kompilera ett låtsasprogram och skicka --verbose switchen till länkaren. Exempelvis **gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded** kommer visa alla filer som framgångsrikt öppnades under länkningen.

Nästa paket att installeras är GCC. Ett exempel angående vad man kan se vid dess körning av **configure** är:

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

Detta är viktigt på grund av de orsaker som nämns ovan. Det demonstrerar även att GCCs configureskript inte använder sökvägarna i PATH för att hitta vilka verktyg som skall användas. När **gcc** väl körs är det dock inte säkert att samma sökvägar, som de vilka returnerades, kommer att användas. För att avgöra vilken länkare som **gcc** kommer använda som standard, kör: **gcc -print-prog-name=ld**.

Detaljerad information kan erhållas från **gcc** genom att skicka med kommandot -v då ett låtsasprogram kompileras. Exempelvis **gcc -v dummy.c** kommer visa detaljerad information om preprocessor, kompilations och assembler stegen, samt även de sökvägar vilka används av **gcc**, samt dessa sökvägars prioritet.

Nästa installation berör sanitized Linux API headers. Dessa möjliggör för det standard C-bibilioteket(Glibc) att dra nytta av funktioner vilka Linux kerneln tillhandahåller.

Nästa paket som installeras är Glibc. De viktigaste aspekterna vid bygget av Glibc är kompilatorn, binära verktyg, samt kernel headers. Kompilatorn är vanligtvis inte några problem, då Glibc alltid kommer använda kompilatorn vilken är relaterad till --host parametern vilken skickas med till dess configure-skript. De binära verktygen och kernel headers kan vara mer komplicerade att bygga. Ta därför inga risker vid bygget av dem, och använd configures tillgängliga flaggor för att säkerställa korrekta val vid konfiguration. Efter att **configure** har körts, analysera innehållet i config.make för att få en insyn i detaljerna. Denna fil återfinns i mappen glibc-build. Notera användningen av CC=”i686-lfs-gnu-gcc” för att kontrollera vilka binära verktyg som används, samt användningen av -nostdinc och -isystem för att kontrollera kompilatorns inkluderade sökväg. Dessa aspekter belyser en viktiga detalj av Glibc-paketet – nämnligen att det är väldigt självständigt när det kommer till sin förmåga att bygga andra paket, och sällan förlitar sig på toolchainstandarder.

Då Binutils byggs en andra gång, kan vi använda oss av --with-lib-path flaggan för att kontrollera bibliotekssökvägen för **ld**.

Då GCC byggs en andra gång, behöver dess sources också modifieras så att GCC instrueras att använda den nya dynamiska länkaren. Att inte göra detta kommer resultera i att GCC-programmen kommer ha namnet på den dynamiska länkaren i värdsystemets /lib mapp inbäddat i sig, vilket förhindrar målet att komma bort från värden och göra så att toolchainen blir självständig. Från denna punkt framåt är toolchainen självtillräcklig och självständig.
De kvarvarande paketen i kapitel 5 bygger alla gentemot det nya Glibc i /tools.

Då chrootmiljön i kapitel 6 äntras, kommer det första större paketet att installeras vara Glibc, till följd av dess ovan mämnda självständiga natur. Då detta paket väl är installerat i /usr, kommer vi att genomföra en kortare konfiguration av toolchainen, och sedan fortsätta med att bygga resten av systemet.

5.3. Generella kompileringsinstruktioner

Vid byggandet av paket finns det ett antal saker och ting värda att ha i bakhuvudet:

- Flertalet av paketen patchas innan kompilering, men endast då patchen behövs för att kringgå ett problem. En patch behövs ofta i både detta och nästa kapitel, men ibland endast i det ena eller andra. Oroa dig därför inte ifall det inte finns instruktioner för installationen av en patch. Varningsmeddelanden om *offset* och *fuzz* kan också uppstå när en patch appliceras. Oroa dig inte p.g.a dessa varningar, då patchen oavsett detta applicerades framgångsrikt.

- Vid kompilering av de flesta paket kommer ett flertal varningar att meddelas. Dessa är normala och kan vanligtvis ignoreras. Dessa varningar är det de verkar vara – varningar om föråldrad, men inte inkorrekt, användning av C och C++ syntaxen. Kodstandard för C uppdateras relativt ofta, och vissa paket använder ännu en äldre standard. Detta är inget problem, men leder till en varning.

- Kolla en sista gång att variabeln `LFS` är korrekt definierad

```
echo $LFS
```

  Säkerställ att den output som returneras leder till den korrekta monteringspunkten för LFS-partitionen.

- Slutligen måste de två punkterna nedan betonas

Viktigt

Alla instruktioner förutsätter att värdsystemets krav, inklusive symboliska länkar, har blivit korrekt definierade:

- **bash** är skalet som används.
- **sh** är en symbolisk länk till **bash**.
- **/usr/bin/awk** är en symbolisk länk till **gawk**.
- **/usr/bin/yacc** är en symbolisk länk till **bison** eller ett skript som exekverar bison.

Viktigt

För att ytterligare betona byggprocessen:

1. Placera alla sources och patchar i en mapp som kommer vara tillgänglig från chroot-miljön. T.ex /mnt/lfs/sources/. Placera inte sources i /mnt/lfs/tools/.
2. Byt till mappen för sources.
3. För varje paket:
   a. Extrahera, med **tar**, paketet som ska byggas. I kapitel 5, säkerställ att du extraherar paketet som användare lfs.
   b. Byt till mappen som skapades då paketet extraherades.
   c. Följ bokens instruktioner för att bygga paketet.
   d. Byt tillbaka till mappen sources.
   e. Ta bort den extraherade mappen sources, ifall inte instruerad att behålla denna.

5.4. Binutils-2.34 - Runda 1

Paketet Binutils innehåller en länkare, en assembler, samt diverse andra verktyg för att hantera objektfiler.

Förväntad byggtid: 1 SBU
Nödvändigt diskutrymme: 625 MB

5.4.1. Installation av Cross Binutils

Kommentar

Gå tillbaka och läs igenom kommentarerna i den föregående sektionen. En god förståelse vad gäller dessa kommentarer kommer leda till att du undviker många problem framöver.

Det är viktigt att Binutils byggs först eftersom både Glibc och GCC genomför diverse test på den tillgängliga länkaren och assemblern, för att avgöra vilka av deras egna funktioner att aktivera.

Binutils egen dokumentation rekommenderar att Binutils byggs i en dedikerad byggmapp:

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

Kommentar

För att värdena på andra pakets SBUer ska vara av nytta, mät tiden vilken det tar att bygga samt installera detta paket. Gör detta genom att infoga kommandona i ett **time** kommando, på detta vis:
time { ./configure ... && ... && make install; }. .

Kommentar

Specifierade SBU-värden samt nödvändigt diskutrymme, innehåller i kapitel 5 inte data för testsviterna.

Förbered nu Binutils för installation:

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

Vad de olika alternativen betyder:

`--prefix=/tools`
Specifierar för configureskriptet att förbereda installation av Binutils i mappen /tools.

`--with-sysroot=$LFS`
Vid cross-kompilering instruerar detta byggsystemet att leta efter nödvändiga målbibliotek i $LFS.

`--with-lib-path=/tools/lib`
Specifierar vilken bibliotekssökväg som länkaren skall vara konfigurerad för att använda.

`--target=$LFS_TGT`
Eftersom maskinbeskrivningen i variabeln LFS_TGT är lite annorlunda än värdet vilket returneras av skriptet **config.guess** kommer denna flagga meddela skriptet **configure** att justera Binutils byggsystem till att bygga en cross-kompilator.

`--disable-nls`
Detta avaktiverar internalisering, eftersom i18n inte är nödvändig för de temporära verktygen.

`--disable-werror`
Detta förhindrar att byggprocessen avstannar ifall kompilatorn rapporterar en varning.

Kompilera paketet.

```
make
```

Kompileringen är nu färdig. Vanligtvis skulle vi köra testsviten, men såhär tidigt i processen är inte ramverken (Tcl, Expect, DejaGNU) tillgängliga ännu. Fördelarna med att köra ett test i detta stadie är även minimala, eftersom de temporära program vilka byggs i detta kapitel snart kommer ersättas med permanenta sådana i kapitel 6.

Ifall du bygger på x86_64, skapa en symlink för att säkerställa toolchainens sammanhållning:

```
case $(uname -m) in
  x86_64) mkdir -v /tools/lib && ln -sv lib /tools/lib64 ;;
esac
```

Installera paketet:

```
make install
```

Detaljer angående detta paket återfinns i sektion 6.18.2.

5.5. GCC-9.2.0 - Pass 1

Detta GCC-paket innehåller GNU kompilatorer, däribland C och C++ kompilatorerna.

Förväntad byggtid: 10 SBU
Nödvändigt diskutrymme: 3.1 GB

5.5.1. Installation av Cross GCC

GCC kräver nu GMP, MPFR och MPC paketen. Eftersom dessa paket kanske inte är inkluderade i din distribution kommer vi att bygga dem med hjälp av GCC. Extrahera varje paket i GCC's source-mapp och byt namn på de mappar vilka skapas, så att GCC automatiskt kommer att använda dem under byggproceduren:

Kommentar

Det är vanligt med missförstånd vad gäller detta kapitel. Proceduren är den examt samma som när det gäller andra paket. Extrahera först tarballen i mappen sources, och byt sedan till mappen vilken skapas. Först efter detta ska du fortsätta med instruktionerna nedan.

```
tar -xf ../mpfr-4.0.2.tar.xz
mv -v mpfr-4.0.2 mpfr
tar -xf ../gmp-6.2.0.tar.xz
mv -v gmp-6.2.0 gmp
tar -xf ../mpc-1.1.0.tar.gz
mv -v mpc-1.1.0 mpc
```

Följance kommando kommer att ändra platsen för GCCs standard dynamiska länkare, för att istället använda den vilken finns i /tools. Det tar även bort /usr/include från GCCs sökvägar. Kör detta kommando:

```
for file in gcc/config/{linux,i386/linux{,64}}.h
do
  cp -uv $file{,.orig}
  sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
      -e 's@/usr@/tools@g' $file.orig > $file
  echo '
#undef STANDARD_STARTFILE_PREFIX_1
#undef STANDARD_STARTFILE_PREFIX_2
#define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done
```

Ifall det är svårt att förstå vad som sker ovan, kan man bryta ned det i mindre steg. Först kopierar vi headerfilerna till filer Med samma namn, men med tillägget ".orig". Det första sed-kommandot lägger sedan till "/tools" till varje instans av "/lib/ld", "/lib64/ld" och "/lib32/ld", medan det andra ersätter hårdkodade instanser av "/usr". Därefter använder vi #define för att flytta standardstartfilens prefix till slutet av denna fil. Notera att det avslutande "/" i "/tools/lib/" är nödvändigt. Slutligen använder vi **touch** för att uppdatera tidsstämplarna på de kopierade filerna. När detta används tillsammans med **cp -u** förhindrar det oväntade förändringar av ursprungsfilerna, ifall kommandona körs flera gånger.

På x86_64 hosts, sätt slutligen namnet för standard-mappen för 64-bit bibliotek till “lib”:

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

GCCs dokumentation rekommenderar att GCC byggs i en dedikerad byggkatalog:

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

Förbered GCC för kompilering:

```
../configure                                       \
    --target=$LFS_TGT                              \
    --prefix=/tools                                \
    --with-glibc-version=2.11                      \
    --with-sysroot=$LFS                            \
    --with-newlib                                  \
    --without-headers                              \
    --with-local-prefix=/tools                     \
    --with-native-system-header-dir=/tools/include \
    --disable-nls                                  \
    --disable-shared                               \
    --disable-multilib                             \
    --disable-decimal-float                        \
    --disable-threads                              \
    --disable-libatomic                            \
    --disable-libgomp                              \
    --disable-libquadmath                          \
    --disable-libssp                               \
    --disable-libvtv                               \
    --disable-libstdcxx                            \
    --enable-languages=c,c++
```

Vad de olika alternativen betyder:

`--with-glibc-version=2.11`
Säkerställer att paketet kommer vara kompatibelt med värdens version av glibc. Det är satt till de minimikrav som är specifierade för Glibc i Host System Requirements.

`--with-newlib`
Eftersom ett fungerande C-bibliotek inte ännu finns tillgängligt, säkerställer detta att konstanten inhibit_libc är definierad då libgcc byggs. Detta förhindrar kompileringen av kod vilken kräver libc.

`--without-headers`
Då en fullständig cross-kompilator skapas, kräver GCC headers vilka är kompatibla med målsystemet. För våra syften är dessa headers inte nödvändiga. Denna flagga ser till att GCC inte letar efter dem.

`--with-local-prefix=/tools`
Det lokala prefixet är platsen där GCC kommer söka efter lokalt installerade includefiler. Standard är /usr /local. att sätta detta till /tools håller värdsystemets /usr /local utanför GCCs sökvägar.

`--with-native-system-header-dir=/tools/include`
Som standard söker GCC igenom /usr /include efter systemheaders. Tillsammans med sysroot flaggan, skulle detta vanligtvis översättas till $LFS/usr/include. Dock, headers vilka kommer installeras i de två nästa sektionerna kommer hamna i $LFS/tools/include. Denna flagga säkerställer att GCC kommer lokalisera dem korrekt. Då GCC byggs en andra gång, kommer denna flagga säkerställa att inga headers från värdsystemet upptäcks.

`--disable-shared`
Denna flagga tvingar GCC att länka dessa interna bibliotek statiskt. Vi gör detta för att undvika problem med värdsystemet.

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

Dessa flaggor avaktiverar stöd för de specifierade biblioteken. Dessa bibliotek kommer inte att kompileras då vi bygger cross-kompilatorn, och är inte nödvändiga för att kunna cross-kompilera det temporära libc.

`--disable-multilib`
På x86_64 stödjer inte LFS ännu multilib konfigurationer. Denna flagga är harmlös för x86.

`--enable-languages=c,c++`
Denna flagga säkerställer att endast kompilatorer för C och C++ byggs. Inga andra kompilatorer behövs ännu.

Kompilera GCC genom att köra:

```
make
```

Kompilering är nu slutförd. Också här skulle det vara bra att köra en testsvit, men som sagt finns inte ramverken för test tillgängliga ännu, och i övrigt gäller tidigare kommentarer vad gäller att köra testsviter såhär tidigt i processen.

Installera paketet:

```
make install
```

Detaljer för detta paket återfinns under sektionen 6.25.2,

5.6. Linux-5.5.3 API Headers

Linux API Headers (i linux-5.5.3.tar.xz) öppnar upp kernelns API för användning av Glibc..

Förväntad byggtid: 0.1 SBU
Nödvändigt diskutrymme: 1 GB

5.6.1. Installation av Linux API Headers

Linux kernel behöver öppna upp ett Application Programming Interface(API) vilket systemets C bibliotek kan använda. Detta görs genom att sanera diverse C headerfiler som medföljer Linux kernelns tarball.

Säkerställ att det inte finns några stale files inbäddade i paketet:

```
make mrproper
```

Extrahera nu de användarsynliga kernel headersen från source. Det rekommenderade make target “headers_install” kan inte användas, eftersom det kräver **rsync**, vilket kanske inte finns tillgängligt. Headersen placeras först i ./usr. och kopieras sedan till den korrekta platsen.

```
make headers
cp -rv usr/include/* /tools/include
```

Detaljer för detta paket återfinns i sektion 6.7.2.

5.7. Glibc-2.31

Paketet Glibc innehåller det huvudsakliga C-biblioteket. Detta bibliotek tillhandahåller de grundläggande rutinerna för att allokera minne, söka igenom mappar, öppna och stänga filer, läsa och skriva filer, stränghantering, och så vidare.

Förväntad byggtid: 4.5 SBU
Nödvändigt disk-utrymme: 896 MB

5.7.1. Installation av Glibc

Glibc dokumentationen rekommenderar att Glibc byggs i en dedikerad byggmapp:

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

Förbered Glibc för kompilering:

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

Vad de olika alternativen betyder:

`--host=$LFS_TGT, --build=$(../scripts/config.guess)`
Den kombinerade effekten av dessa flaggor är att byggsystemet för Glibc konfigurerar sig självt till att cross-kompilera, genom att använda cross-länkaren och cross-kompilatorn i /tools.

`--enable-kernel=3.2`
Specifierar för Glibc att kompilera biblioteket med stöd för 3.2 och senare kernels. Lösningar för äldre kernels är inte aktiverade.

`--with-headers=/tools/include`
Specifierar för Glibc att kompilera sig självt i förhållande till de headers vilka nyligen installerades i mappen /tools. Detta för att Glibc ska veta vilka funktioner som kerneln har, och hur det kan optimisera sig självt gentemot dessa.

Under detta stadie kan det hända att följande varning uppenbarar sig:

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

Det saknade eller inkompatibla **msgfmt** programmet är vanligtvis harmlöst. Detta program är del av paketet Gettext, vilket värddistributionen borde tillhandahålla.

Kommentar

Det har kommit in rapporter om att bygget av detta paket kan misslyckas då man kör en “parallel make”. Om detta inträffar, bygg det igen med flaggan “-j1” aktiverad.

Kompilera paketet:

```
make
```

Installera paketet:

```
make install
```

Varning

Då du når denna punkt, är det viktigt att säkerställa att de grundläggande funktionerna (kompilering och länkning) för den nya toolchainen verkligen fungerar som det är tänkt. Kör följande kommandon:

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

Om allting fungerar som förväntat bör inga errors rapporteras, och output som returneras till det sista kommandot kommer vara av formen:

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

Notera, att för 32bits-maskiner kommer tolkens namn vara:`/tools/lib/ld-linux.so.2`.

Ifall output inte visas som ovan, eller ifall det inte returnerades någon output alls, då är något fel. Efterforska och felsök tidigare steg för att hitta vilket som är felet. Detta är ett problem som måste korrigeras innan det att du fortsätter vidare i boken.

När allt till sist fungerar som tänkt, städa upp med hjälp av:

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

Kommentar

Att bygga Binutils i sektionen efter den nästa kommer fungera som en extra koll att toolchainen verkligen har byggts på ett korrekt sätt och vis. Om Binutils inte byggs korrekt eller alls, är det en indikation på att något har gått fel i den tidigare omgången då Binutils byggdes, eller då GCC eller Glibc byggdes.

Detaljer för detta paket återfinns i sektion 6.9.3.

5.8. Libstdc++ from GCC-9.2.0

Libstdc++ är standardbiblioteket för C++. Det behövs för att kompilera C++ kod (delar av GCC är skrivet i C++), men vi behövde skjuta upp installationen av detta paket då vi byggde gcc i första rundan, eftersom det kräver Glibc.

Förväntad byggtid: 0.5 SBU
Nödvändigt disk-utrymme: 878 MB

5.8.1. Installation av Target Libstdc++

Kommentar

Libstdc++ är del av GCC sources. Du bör först extrahera GCC tarballen och byta till gcc-9.2.0 mappen.

Skapa en separat byggmapp för GCC-9.2.0, och gå in i den:

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

Förbered Libstdc++ för kompilering:

```
../libstdc++-v3/configure           \
    --host=$LFS_TGT                 \
    --prefix=/tools                 \
    --disable-multilib              \
    --disable-nls                   \
    --disable-libstdcxx-threads     \
    --disable-libstdcxx-pch         \
    --with-gxx-include-dir=/tools/$LFS_TGT/include/c++/9.2.0
```

Vad de olika alternativen betyder:

`--host=...`
Indikerar att cross-kompilatorerna vi just har byggt ska användas, och inte de i `/usr/bin`

`--disable-libstdcxx-threads`
Eftersom första rundan av gcc är byggt utan stöd för threads, kan inte C++ bibliotekt för threads byggas heller.

`--disable-libstdcxx-pch`
Denna flagga motverkar installationen av förkompilerade include filer, vilka inte behövs just nu.

`--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/9.2.0`
Detta är platsen där C++ kompilatorn söker efter include filer. I en normal byggnation skickas denna information till Libstdc++ **configure**s alternativ per automatik. I vårt fall måste denna information explicit specifieras.

Kompilera libstdc++ genom att köra:

```
make
```

Installera biblioteket:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.25.2.

5.9. Binutils-2.34 - Pass 2

Paketet Binutils innehåller en länkare, en assembler, samt andra verktyg för att hantera objektfiler.

Förväntad byggtid: 1.1 SBU
Nödvändigt disk-utrymme: 651 MB

5.9.1. Installation of Binutils

Skapa återigen en separat byggmapp.

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

Förbered Binutils för kompilation:

```
CC=$LFS_TGT-gcc                \
AR=$LFS_TGT-ar                 \
RANLIB=$LFS_TGT-ranlib         \
../configure                   \
    --prefix=/tools            \
    --disable-nls              \
    --disable-werror           \
    --with-lib-path=/tools/lib \
    --with-sysroot
```

Vad de olika alternativen betyder:

`CC=$LFS_TGT-gcc AR=$LFS_TGT-ar RANLIB=$LFS_TGT-ranlib`
Eftersom detta är en ursprunglig byggnation av Binutils, säkerställer dessa variabler att byggsystemet använder cross-kompilatorn samt relaterade verktyg, istället för de som redan finns på värdsystemet.

`--with-lib-path=/tools/lib`
Leder till att configure-skriptet specifierar sökvägen till bibiloteken under kompileringen av Binutils, vilket resulterar i att /tools /lib skickas till länkaren. Detta förhindrar länkaren från att söka igenom biblioteksmappar på värden.

`--with-sysroot`
Definierar en standard (icke-existent) mapp för sysroot: /tools/$LFS_TGT/sys-root. Den är användbar när man letar efter ett delat objekt vilket krävs av andra delade objekt explicit inkluderade i länkarens kommando. Dessa objekt eftersöks i mapparna listade i <sysroot>/etc/ld.so.conf, och om detta misslyckas i länkarens sökvägar. Om denna flagga inte är satt så används värdens /etc /ld.so.conf – program som existerar på värden kommer alltså att användas, vilket är något vi försöker att undvika.

Kompilera paketet:

```
make
```

Installera paketet:

```
make install
```

Förbered nu länkaren för nästa kapitels omjusteringsfas:

```
make -C ld clean
make -C ld LIB_PATH=/usr/lib:/lib
cp -v ld/ld-new /tools/bin
```

Vad de olika make-parametrarna betyder:

`-C ld clean`
Specifierar för make att ta bort alla kompilerade filer i undermappen “ld”.

`-C ld LIB_PATH=/usr/lib:/lib`
Återbygger allt i undermappen “ld”. Genom att via kommandolinjen specifiera LIB_PATH variabeln för makefile, möjliggör det för oss att skriva över standardvärdet för de temporära verktygen och peka detta till den korrekta slutgiltiga platsen. Denna variabel specifierar länkarens standard sökväg. Detta kommer till nytta i nästa kapitel.

Detaljer för detta paket återfinns i sektion 6.18.2.

5.10. GCC-9.2.0 - Pass 2

Paketet GCC innehåller kompilatorer för GNU. Dessa inkluderar kompilatorer för C samt C++.

Förväntad byggtid: 13 SBU
Nödvändigt disk-utrymme: 3.7 GB

5.10.1. Installation av GCC

Vår första byggnation av GCC har installerat ett antal interna systemheaders. Vanligtvis kommer en av dem, limits.h, att inkludera det motsvarande systemets limits.h header, i detta fall /tools /include /limits.h. Dock, då den första GCC versionen byggdes, existerade inte denna fil, så den interna header vilken GCC installerade är en partiell fil vilken inte tillhandahåller de utökade funktioner vilka återfinns i systemheadern. Detta var lämpligt vid bygge av det temporära Libc, men denna variant av GCC kräver den kompletta interna systemheadern. Skapa en komplett version av denna interna header genom att använda ett kommando som är identiskt gentemot vad GCCs byggsystem vanligtvis använder:

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

Återigen, ändra platsen för var GCC som standard söker efter sin dynamiska länkare, till: `/tools` .

```
for file in gcc/config/{linux,i386/linux{,64}}.h
do
  cp -uv $file{,.orig}
  sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
      -e 's@/usr@/tools@g' $file.orig > $file
  echo '
#undef STANDARD_STARTFILE_PREFIX_1
#undef STANDARD_STARTFILE_PREFIX_2
#define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done
```

Om du bygger på x86_64, ändra standard mappnamnet för 64bits-bibliotek till “lib”:

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

Även här kräver GCC-paketet GMP, MPFR och MPC paketen. Extrahera tarballsen och förflytta dem till de korrekta mapparna:

```
tar -xf ../mpfr-4.0.2.tar.xz
mv -v mpfr-4.0.2 mpfr
tar -xf ../gmp-6.2.0.tar.xz
mv -v gmp-6.2.0 gmp
tar -xf ../mpc-1.1.0.tar.gz
mv -v mpc-1.1.0 mpc
```

Fixa nu ett problem som introducerades med Glibc-2.31:

```
sed -e '1161 s|^|//|' \
    -i libsanitizer/sanitizer_common/sanitizer_platform_limits_posix.cc
```

Skapa en separat byggmapp igen:

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

Innan GCC byggs, kom ihåg att avaktivera alla miljövariabler vilka prioriteras över de optimerande standardflaggorna.

Förbered nu GCC för kompilering:

```
CC=$LFS_TGT-gcc                                         \
CXX=$LFS_TGT-g++                                        \
AR=$LFS_TGT-ar                                          \
RANLIB=$LFS_TGT-ranlib                                  \
../configure                                            \
    --prefix=/tools                                     \
    --with-local-prefix=/tools                          \
    --with-native-system-header-dir=/tools/include      \
    --enable-languages=c,c++                            \
    --disable-libstdcxx-pch                             \
    --disable-multilib                                  \
    --disable-bootstrap                                 \
    --disable-libgomp
```

Vad de olika alternativen betyder:

`--enable-languages=c,c++`
Säkerställer att både C och C++ kompilatorerna byggs.

`--disable-libstdcxx-pch`
Bygg inte den förkompilerade headern (PCH) för libstdc++. Den är stor, och vi har ingen nytta av den.

`--disable-bootstrap`
För urspungliga byggnationer av GCC, är standard att skapa en “bootstrap” variant. Det innebär att GCC inte bara kompileras, utan kompileras många gånger. Det använder programmen kompilerade i första rundan för att kompilera sig själv i en andra sådan, och sedan en tredje gång. De andra och tredje iterationerna jämförs med varandra, för att säkerställa att kompilatorn kan reproducera sig själv på ett exakt vis. Detta antyder även att kompileringen var framgångsrik. Detta är dock inte nödvändigt, då metoden för att bygga LFS tillhandahåller en solid kompilator.

Compile the package:

```
make
```

Installera paketet:

```
make install
```

Skapa slutligen en symlink. Många program och skript kör **cc** istället för **gcc**, vilket beror på att program då kan hållas generiska och därför kan användas på UNIX-system där GNUs C-kompilator inte är installerad. Att köra **cc** innebär att systemadministratören kan välja vilken C-kompilator som skall installeras.

```
ln -sv gcc /tools/bin/cc
```

Caution

Då du når denna punkt, är det viktigt att säkerställa att de grundläggande funktionerna (kompilering och länkning) hos den nya toolchainen verkligen fungerar som det är tänkt. Kör följande kontrollkommando:

```
echo 'int main(){}' > dummy.c
cc dummy.c
readelf -l a.out | grep ': /tools'
```

Om allting fungerar som förväntat bör inga errors rapporteras, och output som returneras till det sista kommandot kommer vara av formen:

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

Notera, att för 32bits-maskiner kommer tolkens namn vara:/tools/lib/ld-linux.so.2.

Ifall output inte visas som ovan, eller ifall det inte returnerades någon output alls, då är något fel. Efterforska och felsök tidigare steg för att hitta vilket som är felet. Detta är ett problem som måste korrigeras innan det att du fortsätter vidare i boken. Kör först kontrollkommandot igen, men använd **gcc** istället för **cc**. Säkerställ att sökvägen är correct genom att köra **echo $PATH.** Om path är inkorrekt kan det betyda att du inte är inloggad som användaren lfs, eller att något gick fel i steg 4.4.

När allt till sist fungerar som tänkt, städa upp med hjälp av:

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

Detaljer för detta paket återfinns i sektion 6.25.2.

5.11. Tcl-8.6.10

Paketet TCL innehåller Tool Command Language.

Förväntad byggtid: 0.9 SBU
Nödvändigt disk-utrymme: 72 MB

5.11.1. Installation av Tcl

Detta paket samt de följande två (Expect and DejaGNU) installeras för att understödja körandet av testsviterna för GCC, Binutils, samt diverse andra paket. Att installera alla dessa tre paket i syfte att kunna köra test kanske verkar överdrivet, men det är väldigt betryggande att veta att de viktigaste verktygen fungerar korrekt. Även ifall testsviterna inte körs i detta kapitel, är dessa paket nödvändiga för att köra testsviterna i kapitel 6.

Notera att TCL-paketet som används här är en minimal version, som dock klarar av att köra LFS-testen. Ifall du är intresserad av det kompletta paketet, hänvisas du till: BLFS TCL procedures.

Förbered TCL för kompilering:

```
cd unix
./configure --prefix=/tools
```

Bygg paketet:

```
make
```

Kompilering är nu färdig.
Det är inte nödvändigt att köra testen i detta kapitel, men om du ändå önskar göra detta:

```
TZ=UTC make test
```

Testsviten för TCL kan stöta på problem under vissa värdförhållanden. Testmisslyckanden är därför inte förvånande, och skall inte betraktas som kritiska. Parametern TZ=UTC sätter tids-zonen till Coordinated Universal Time(UTC), men endast under den tid som testsviten körs. Detta säkerställer att klocktesten körs i en korrekt kontext. Detaljer Vad gäller TZ miljövariabeln, återfinns i kapitel 7.

Installera paketet:

```
make install
```

Gör det installerade biblioteket skrivbart, så att felsökningssymboler senare kan tas bort:

```
chmod -v u+w /tools/lib/libtcl8.6.so
```

Installera TCLs headers. Nästa paket, Expect, kräven dem i syfte att byggas.

```
make install-private-headers
```

Skapa nu den nödvändiga symboliska länken:

```
ln -sv tclsh8.6 /tools/bin/tclsh
```

5.11.2. Innehåll Tcl

Installerade program: tclsh (link to tclsh8.6) and tclsh8.6
Installerat bibliotek: libtcl8.6.so, libtclstub8.6.a

Korta beskrivningar

tclsh8.6	Tcl’s kommandoskal
tclsh	En länk till tclsh8.6
`libtcl8.6.so`	Tcl’s bibliotek
`libtclstub8.6.a`	Tcl’s Stub bibliotek

5.12. Expect-5.45.4

Paketet Expect innehåller ett program vilket möjliggör att skriptade dialoger kan köras med andra interaktiva program.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 4.0 MB

5.12.1. Installation of Expect

Tvinga först Expect's configure-skript att använda /bin /stty istället för /usr /local /bin /stty vilken kan finnas på värdens system. Detta säkerställer vår toolchains sammanhållning.

```
cp -v configure{,.orig}
sed 's:/usr/local/bin:/bin:' configure.orig > configure
```

Förbered Expect för kompilering:

```
./configure --prefix=/tools       \
            --with-tcl=/tools/lib \
            --with-tclinclude=/tools/include
```

Vad de olika alternativen betyder:

`--with-tcl=/tools/lib`
Säkerställer att configure-skriptet hittar Tcl-installationen i mappen för temporära verktyg, istället för att leta i värdens mappar.

`--with-tclinclude=/tools/include`
Specifierar uttryckligen för Expect var det går att hitta Tcl's interna headers. Användandet av denna flagga innebär undvikandet av omständigheter där **configure** misslyckas eftersom Tcl's headers inte kan automatiskt upptäckas.

Bygg paketet:

```
make
```

Kompileringen är nu slutförd.
Om du vill köra testsviten:

```
make test
```

Denna testsvit är känd för att returna fel relaterade till värdsystemet. Betrakta därför inte fel i detta test som kritiska.

Installera paketet:

```
make SCRIPTS="" install
```

Vad detta alternativ betyder:

`SCRIPTS=""`
Förhindrar installationen av de extra Expect-skripten, vilka inte är nödvändiga.

5.12.2. Contents of Expect

Installerat program: expect
Installerat bibliotek: libexpect-5.45.so

Korta beskrivningar

expect	Kommunicerar med andra interaktiva program, genom användandet av ett skript
`libexpect-5.45.so`	Innehåller funktioner vilka möjliggör för Expect att användas som en Tcl-förlängning, eller direkt från C eller C++(utan Tcl).

5.13. DejaGNU-1.6.2

Paketet DejaGNU innehåller ett ramverk för att testa andra program.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 3.2 MB

5.13.1. Installation av DejaGNU

Förbered:

```
./configure --prefix=/tools
```

Bygg och installera:

```
make install
```

För att testa resultat:

```
make check
```

5.13.2. Innehåll DejaGNU

Installerade program: runtest

Korta beskrivningar

runtest Ett wrapper-skript vilket lokaliserar det korrekt **expect** skalet och sedan köra DejaGNU.

5.14. M4-1.4.18

Paketet M4 innehåller en makro-processor.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 20 MB

5.14.1. Installation av M4

Fixa först några omständigheter som introducerades med glibc-2.28:

```
sed -i 's/IO_ftrylockfile/IO_EOF_SEEN/' lib/*.c
echo "#define _IO_IN_BACKUP 0x100" >> lib/stdio-impl.h
```

Förbered:

```
./configure --prefix=/tools
```

Kompilera:

```
make
```

Kompilering är nu färdig. Kör test om du vill.

```
make check
```

Installera:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.16.2.

5.15. Ncurses-6.2

Paketet Ncurses innehåller bibliotek för terminal-självständig hantering av karaktärs-skärmar.

Förväntad byggtid: 0.6 SBU
Nödvändigt disk-utrymme: 41 MB

5.15.1. Installation av Ncurses

Säkerställ att **gawk** finns tillgängligt:

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

Förbered:

```
./configure --prefix=/tools \
            --with-shared   \
            --without-debug \
            --without-ada   \
            --enable-widec  \
            --enable-overwrite
```

Vad de olika alternativen betyder:

`--without-ada`
Säkerställer att Ncurses inte bygger stöd för Ada-kompilatorn, vilken kan finnas tillgänglig på värden men som inte kommer vara tillgänglig i **chroot** miljön.

`--enable-overwrite`
Specifierar för Ncurses att installera dess headerfiler i /tools /include istället för i /tools /include /ncurses, detta görs för att säkerställa att andra paket kan hitta Ncurses headers.

`--enable-widec`
Denna flagga leder till att wide-character(libncursesw.so.6.2) bibliotek byggs, istället för normala sådana (t.ex libncurses.so.6.2). Dessa wide-character bibliotel kan vanligtvis användas i både multibyte och traditionella 8-bit omständigheter, medan vanliga bibliotek endast skulle fungera som önskat i 8bits omständigheterna. Wide-character och normala bibliotek är source-kompatibla, men inte binär-kompatibla.

Kompilera:

```
make
```

Detta pakets testsvit kan endast köras efter att paketet installerats. Testen finns i mappen test/. Läs README för instruktioner.

Installera:

```
make install
ln -s libncursesw.so /tools/lib/libncurses.so
```

Detaljer för detta paket återfinns i sektion 6.27.2.

5.16. Bash-5.0

Paketet Bash innehåller Bourne-Again SHell.

Förväntad byggtid: 0.4 SBU
Nödvändigt disk-utrymme: 67 MB

5.16.1. Installation av Bash

Förbered:

```
./configure --prefix=/tools --without-bash-malloc
```

Vad alternativen betyder:

`--without-bash-malloc`
Stänger av Bash's minnesallokering, vilken kan leda till skapandet av segmentationsfel. Att stänga av denna funktion leder till att Bash använder sig av Glibc's malloc-funktioner. Dessa är mer stabila.

Kompilera:

```
make
```

Kompilering är färdig. Kör test om du vill.

```
make tests
```

Installera:

```
make install
```

Skapa en länk för de program som använder **sh** som sitt skal:

```
ln -sv bash /tools/bin/sh
```

Detaljer för detta paket återfinns i sektion 6.35.2.

5.17. Bison-3.5.2

Paketet Bison innehåller en parser generator.

Förväntad byggtid: 0.3 SBU
Nödvändigt disk-utrymme: 43 MB

5.17.1. Installation av Bison

Förbered:

```
./configure --prefix=/tools
```

Kompilera:

```
make
```

För att köra testen:

```
make check
```

Installera paketet:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.32.2.

5.18. Bzip2-1.0.8

Paketet Bzip2 innehåller program för att komprimera samt dekomprimera filer. Att komprimera filer med **bzip2** leder till mycket bättre resultat än då man använder det traditionella **gzip**.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 6.4 MB

5.18.1. Installation av Bzip2

Paketet Bzip2 innehåller inte ett **configure-sk**ript. Istället finns det två make-filer, en för det delade biblioteket, och en för det statiska biblioteket. Eftersom vi behöver båda två, genomför vi kompileringen i två steg. Först det delade:

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

Vad alternativen betyder:

`-f Makefile-libbz2_so`
Detta leder till att bzip2 byggs med hjälp av en annan make-fil. i detta fall Makefile-libbz2_so, vilken skapar ett dynamiskt libbz2.so bibliotek, samt länkar Bzip2's verktyg gentom det.

Kompilera och testa paketet med:

```
make
```

Installera:

```
make PREFIX=/tools install
cp -v bzip2-shared /tools/bin/bzip2
cp -av libbz2.so* /tools/lib
ln -sv libbz2.so.1.0 /tools/lib/libbz2.so
```

Detaljer för detta paket återfinns i sektion 6.12.2.

5.19. Coreutils-8.31

Paketet Coreutils innehåller verktyg för att visa och konfigurera systemes grundläggande karaktäristik.

Förväntad byggtid: 0.7 SBU
Nödvändigt disk-utrymme: 157 MB

5.19.1. Installation av Coreutils

Förbered:

```
./configure --prefix=/tools --enable-install-program=hostname
```

Vad alternativen betyder:

`--enable-install-program=hostname`
Detta möjliggör för **hostname** binären att byggas och installeras – detta alternativ är avaktiverat som standard, men krävs för att köra Perl's testsvit.

Kompilera:

```
make
```

Kompilering är nu färdig. Kör testsviten om du vill.

```
make RUN_EXPENSIVE_TESTS=yes check
```

Valet *`RUN_EXPENSIVE_TESTS=yes`* Specifierar för testsviten att köra ett antal extra test, vilka anses vara, i förhållande till CPU och minne, kostsamma på vissa plattformar. De är vanligtvis inget problem på Linux dock.

Installera:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.54.2.

5.20. Diffutils-3.7

Paketet Diffutils innehåller program vilka visar skillnaden emellan olika filer och mappar.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 26 MB

5.20.1. Installation av Diffutils

Förbered:

```
./configure --prefix=/tools
```

Kompilera:

```
make
```

Kompilering färdig.
Kör test-svit om du vill:

```
make check
```

Installera:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.56.2.

5.21. File-5.38

Paketet File innehåller ett verktyg för att identifiera vad för typ av fil en eller flera filer är.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 20 MB

5.21.1. Installation av File

Förbered:

```
./configure --prefix=/tools
```

Kompilera:

```
make
```

Kompilering är nu färdig.
Kör test-sviten om du vill:

```
make check
```

Installera:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.14.2.

5.22. Findutils-4.7.0

Paketet Findutils innehåller program för att finna filer. Dessa program söker rekursivt igenom en mapphierarki, samt kan även användas för att skapa, underhålla, samt genomsöka en databas (ofta snabbare än den rekursiva sökningen, men opålitlig ifall databasen inte har blivit uppdaterad nyligen).

Förväntad byggtid: 0.3 SBU
Nödvändigt disk-utrymme: 39 MB

5.22.1. Installation av Findutils

Förberedelser:

```
./configure --prefix=/tools
```

Kompilera:

```
make
```

Kompilering färdig.
Kör test-svit om du vill:

```
make check
```

Installera:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.58.2.

5.23. Gawk-5.0.1

Paketet Gawk innehåller program för att manipulera text-filer.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 46 MB

5.23.1. Installation av Gawk

Förberedelser:

```
./configure --prefix=/tools
```

Kompilera:

```
make
```

Kompilering färdig.
Kör test-sviten om du vill:

```
make check
```

Installera:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.57.2.

5.24. Gettext-0.20.1

Paketet Gettext package innehåller verktyg för internalisering samt lokalization. Detta möjliggör för program att kompileras med hjälp av NLS (Native Language Support), vilket tillåter dem att skicka output i sina ursprungliga språk.

Förväntad byggtid: 1.6 SBU
Nödvändigt disk-utrymme: 300 MB

5.24.1. Installation av Gettext

För den temporära toolchainen behöver vi endast installer tre av programmen från Gettext.

Förberedelser:

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

Vad alternativen betyder:

`--disable-shared`
Vi behöver inte installera några av Gettext's delade bibliotek just nu, därför behöver vi inte heller bygga dem.

Kompilering:

```
make
```

Pågrund av den begränsade miljön, rekommenderas det inte att köra test-sviten just nu.

Installera programmen **msgfmt**, **msgmerge** och **xgettext**:

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

Detaljer för detta paket återfinns i sektion 6.47.2.

5.25. Grep-3.4

Paketet Grep tillhandahåller program för att söka igenom filer.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 25 MB

5.25.1. Installation av Grep

Förberedelser:

```
./configure --prefix=/tools
```

Kompilering:

```
make
```

Kompilering färdig.
Kör test-sviten om du vill:

```
make check
```

Installation:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.34.2.

5.26. Gzip-1.10

Paketet Gzip innehåller program för att komprimera samt dekomprimera filer.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 10 MB

5.26.1. Installation av Gzip

Förberedelser:

```
./configure --prefix=/tools
```

Kompilering:

```
make
```

Kompilering färdig.
Kör test-sviten om du vill:

```
make check
```

Installation:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.62.2.

5.27. Make-4.3

Paketet Make innehåller ett program för att kompilera paket.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 16 MB

5.27.1. Installation av Make

Förberedelser:

```
./configure --prefix=/tools --without-guile
```

Vad alternativen betyder:

`--without-guile`
Säkerställer att Make inte länkas gentemot Guile-bibliotek, vilka kan finnas tillgängliga på värd-systemet men inte kommer vara tillgängliga i nästa kapitels **chroot**-miljö.

Kompilering:

```
make
```

Kompilering färdig.
Kör test-sviten om du vill:

```
make check
```

Installation:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.67.2.

5.28. Patch-2.7.6

Paketet Patch innehåller ett program för att modifiera och skapa filer genom att applicera en “patch”-fil, vanligtvis skapad av programmet **diff**.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 13 MB

5.28.1. Installation av Patch

Förberedelser:

```
./configure --prefix=/tools
```

Kompilering:

```
make
```

Kompilering färdig.
Kör test-sviten om du vill:

```
make check
```

Installation:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.68.2.

5.29. Perl-5.30.1

Paketet Perl innehåller språket Practical Extraction and Report.

Förväntad byggtid: 1.5 SBU
Nödvändigt disk-utrymme: 275 MB

5.29.1. Installation av Perl

Förberedelser:

```
sh Configure -des -Dprefix=/tools -Dlibs=-lm -Uloclibpth -Ulocincpth
```

Vad alternativen betyder:

-des
Detta är en kombination av tre alternativ: -d använder standardinställningar för alla aspekter. -e säkerställer att alla uppgifter slutförs. -s tystar output vilken inte är absolut nödvändig.

-Uloclibpth and *-Ulocincpth*
Dessa tillval avdefinierar variabler vilka kan leda till att konfigurationen söker efter lokalt installerade komponenter vilka existerar på värdsystemet.

Bygg paketet:

```
make
```

Även om Perl har en test-svit, är det bättre att vänta med att köra denna tills i nästa kapitel.

Bara ett mindre antal av verktygen och biblioteken behöver installeras just nu:

```
cp -v perl cpan/podlators/scripts/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.30.1
cp -Rv lib/* /tools/lib/perl5/5.30.1
```

Detaljer för detta paket återfinns i sektion 6.41.2.

5.30. Python-3.8.1

Paketet Python3 innehåller python's utvecklingsmiljö. Språket är användbart för objektorienterad programmering, skriva skript, skapa prototyper av större program, eller för att utveckla fullständiga applikationer.

Förväntad byggtid: 1.3 SBU
Nödvändigt disk-utrymme: 409 MB

5.30.1. Installation av Python

Kommentar

Det finns två paket-filer vilkas namn börjar med "python". Den som skall extraheras är Python-3.8.1.tar.xz (uppmärksamma den stora första bokstaven).

Detta paket bygger först Pythons tolk, sedan ett antal av standard-modulerna. Det huvudsakliga skript som används för att bygga modulerna är skrivet i Python, och använder hårdkodade sökvägar till värdens /usr /include same /usr /lib. För att förhindra dem från att användas, kör:

```
sed -i '/def add_multiarch_paths/a \        return' setup.py
```

Förberedelser:

```
./configure --prefix=/tools --without-ensurepip
```

Vad alternativen betyder:

`--without-ensurepip`
Denna flagga avaktiverar Pythons paketinstallerare, vilken inte behövs just nu.

Kompilering:

```
make
```

Kompilering färdig. Test-sviten kräven TK och X Windows, och kan därför inte köras just nu.

Installation:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.51.2.

5.31. Sed-4.8

Paketet Sed innehåller en stream-editor.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 21 MB

5.31.1. Installation av Sed

Förberedelser:

```
./configure --prefix=/tools
```

Kompilering:

```
make
```

Kompilering färdig.
Kör test-sviten om du vill:

```
make check
```

Installation:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.29.2.

5.32. Tar-1.32

Paketet Tar innehåller ett arkiverings-program.

Förväntad byggtid: 0.3 SBU
Nödvändigt disk-utrymme: 38 MB

5.32.1. Installation av Tar

Förberedelser:

```
./configure --prefix=/tools
```

Kompilering:

```
make
```

Kompilering färdig.
Kör test-sviten om du vill:

```
make check
```

Install the package:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.70.2.

5.33. Texinfo-6.7

Paketet Texinfo innehåller program för att läsa, skriva, samt konvertera info-sidor.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 104 MB

5.33.1. Installation av Texinfo

Förberedelser:

```
./configure --prefix=/tools
```

Kommentar

Som del av konfigurerings-processen körs ett test vilket indikerar ett error för TestXS_la-TestXS.lo. Detta är inte relevant för LFS och bör ignoreras.

Kompilering:

```
make
```

Kompilering färdig.
Kör test-sviten om du vill:

```
make check
```

Installation:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.71.2.

5.34. Xz-5.2.4

Paketet Xz innehåller program för att komprimera samt dekomprimera filer. Det tillhandahåller möjligheter för formaten Izma samt det nyare Xz. Att komprimera filer med hjälp av **xz** leder till en bätte komprimerings-procent än med de traditionella **gzip** och **bzip2** kommandona.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 18 MB

5.34.1. Installation av Xz

Förberedelser:

```
./configure --prefix=/tools
```

Kompilering:

```
make
```

Kompilering färdig.
Kör test-sviten om du vill:

```
make check
```

Installation:

```
make install
```

Detaljer för detta paket återfinns i sektion 6.13.2.

5.35. Minimalisering

Stegen i denna sektion är frivilliga, men ifall LFS-partitionen är relativt liten, är det fördelaktigt att veta att vissa delar kan tas bort. De körbara filerna samt biblioteken innehåller tillsvidare ca 70MB felsöknings-symboler. Ta bort dessa symboler med:

```
strip --strip-debug /tools/lib/*
/usr/bin/strip --strip-unneeded /tools/{,s}bin/*
```

Dessa kommandon kommer att hoppa över ett antal filer, samt rapportera att deras filformat inte kan kännas igen. De flesta av dessa är skript och inte binärer. Använd även systemets strip-kommando för att inkludera strip-binären i /tools.

Säkerställ att du *inte* använder --strip-unneeded på paketen. De statiska biblioteken skulle då förstöras och toolchainens paket skulle behöva byggas om.

För att spara mer utrymme, ta bort dokumentationen:

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

Ta bort oväsentliga filer:

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

Du borde nu ha åtminstone 3GB ledigt utrymme i $LFS, vilket i nästa fas kan användas för att bygga och installera Glibc och GCC. Ifall du kommer kunna bygga Glibc, kommer du kunna bygga resten av paketen också.

5.36. Byte av ägarskap

Kommentar

Resten av kommandona i denna bok måste köras som användaren root, och inte som användaren lfs. Dubbelkolla även så att $LFS är definierad för root's miljö.

För tillfället så ägs mappen $LFS/tools av användaren lfs, vilket är en användare som endast existerar på värdsystemet. Ifall mappen $LFS/tools behålls som den är, kommer filerna i denna mapp ägas av ett användar-ID som inte har ett motsvarande konto. Detta är farligt, för ett användarkonto som skapas senare kan få samma användar-ID och därigenom tilldelas ägarskap över mappen, med potentiella åtföljande konsekvenser för systemets säkerhet.

För att undvika detta, så kan du lägga till användaren lfs till det nya LFS-systemet då du skapar filen /etc/passwd, om du ser till att tillskriva den samma användare och gruppid som på värdsystemet. Det är dock bättre att ge root ägandet:

```
chown -R root:root $LFS/tools
```

Även ifall $LFS/tools kan tas bort efter det att LFS-systemet väl har färdigställts, så kan den även behållas i syfte att bygga fler system av samma ***bokversion***. Hur $LFS/tools bäst säkerhetskopieras är upp till var och en att avgöra.

Var uppmärksam

Om du planerar att behålla de temporära verktygen för att bygga framtida LFS-system, så är det nu dags att säkerhetskopiera dem. De kommandon vilka följer i kapitel 6 kommer att förändra verktygen och göra dem värdelösa för syftet att bygga framtida system.

Kapitel III. Att bygga LFS-systemet

Chapter 6. Installation av grundläggande system-mjukvara

6.1. Introduktion

I detta kapitel börjar vi bygga LFS-systemet på riktigt. Med detta menas att vi använder **chroot** för att ta oss in i det temporära och minimala Linux-systemet, förbereder ett antal saker, och sedan börjar installera paketen.

Processen för att installera mjukvaran är enkel. Även om i många fall instruktionerna kunde göras kortare och mer generiska, har vi tillhandahållit de fulla instruktionerna för varje paket, i syfte att minimerar risken för misstag. Nyckeln till att förstå hur ett Linux-system fungerar, är att förstå vad de olika paketen gör och hur de används.

Vi rekommenderar inte att använda optimiseringarna. De kan få ett paket att drivas snabbare, men de kan också leda till kompileringssvårigheter, samt problem då programmen körs. Om ett paket vägrar att kompileras då optimisering används, gör om steget utan optimiseringen. Även ifall paketet kompileras vid användning av optimisering, finns ändå risken att det har kompilerats inkorrekt till följd av de komplicerade förhållandena emellan kod och byggverktyg. Notera även att värden för -march och -mtune vilka inte specifieras i boken, inte har blivit testade. Detta kan leda till problem med toolchain-paketen(Binutils. GCC, Glibc). De begränsade potentiella fördelarna som kan tillfredställas av kompilatoroptimisering vägs ofta ut av riskerna. De användare som bygger LFS en första gång, rekommenderas att bygga utan optimisering. Det slutliga systemet kommer oavsett vara både väldigt snabbt samt stabilt.

Ordningen i vilka paket installeras behöver följas strikt, för att säkerställa att inget program av misstag får en sökväg till /tools hårdkodad in i sig. Av samma anledning ska inte separata paket kompileras parallelt med varandra. Det kan spara tid, men kan resultera i ett program som har en hårdkodad sökväg till /tools, vilket kommer leda till att detta program slutar fungera då denna mapp tas bort.

Innan instruktionerna för varje paket-installation, specifieras information om paketet. Detta inkluderar en kort beskrivning angående vad det innehåller, hur lång tid det kommer ta att bygga, samt hur mycket diskutrymme som krävs under själva processen. Efter installationen följer listor med program och bibliotek vilka paketet installerar.

Kommentar

SBU-värdena samt nödvändigt diskutrymme inkluderar även kraven för test-sviterna.

6.1.1. Angående bibliotek

Rent generellt avråds läsaren från att bygga och installera statiska bibliotek. Det ursprungliga syftet med de flesta sådana paket har föråldrats i takt med att moderna Linux-system växt fram. Att länka ett statiskt bibliotek in i ett program kan även vara skadligt. Om en uppdatering behöver appliceras till biblioteket, måste alla program vilka använder paketet länkas om till det nya sådana. Eftersom det inte alltid är uppenbart att statiska bibliotek används, kanske de relevanta programmen(samt procedurerna nödvändiga för själva länkningen) inte kommer vara kända.

I procedurerna i kapitel 6, tar vi bort eller avaktiverar installationen av de flesta statiska biblioteken. Detta görs vanligtvis genom att flaggan --disable-static skickas till **configure. i** andra fall kan andra metoder behöva användas. I ett antal fall, särskilt vad gäller glibc och gcc, kommer användningen av statiska bibliotek vara absolut nödvändig.

För en mer komplett diskussion vad gäller bibliotek, hänvisas läsaren till *Libraries: Static or Shared* i boken BLFS.

6.2. Förberedelser av den virtuella kernelns filsystem

Diverse filsystem vilka exporteras av kerneln används för att kommunicera till och från kerneln. Dessa filsystem är virtuella i sådan mån att inget diskutrymme krävs för dem. Filsystemens innehåll lagras i (ram?)minnet.

Börja med att skapa mappar genom vilka filsystemen kan monteras:

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

6.2.1. Skapandet av enhets-noder

När kerneln startar upp systemet, kräver den närvaron av ett antal enhets-noder, och då särskilt enheterna *console* och *null*. Enhetsnoderna måste skapas på hårddisken så att de är tillgängliga innan det att **udev** har startats, och ytterligare då Linux startas med hjälp av init=/bin/bash. Skapa de nödvändiga enheterna med följande kommandon:

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

6.2.2. Montering och hantering av /dev

Den rekommenderar metoden för att befolka /dev med enheter, är att montera ett virtuellt filsystem (likt tmpfs) till mappen /dev, och tillåta att enheterna skapas dynamiskt på detta virtuella filsystem allt eftersom de upptäcks eller det sker tillgång till dem. Skapandet av enheter sker vanligtvis vid uppstart, av udev. Eftersom detta nya system inte ännu har Udev och inte heller ännu startats upp, är det nödvändigt att monterna och befolka /dev manuellt. Detta görs igenom att Bind mounting av värdsystemets mapp /dev. Bind mount är ett speciellt sorts montering vilken tillåter att man skapar en spegelbild av en mapp eller monteringspunkt vid någon anna plats. Används följande kommando för detta:

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

6.2.3. Montering av Virtual Kernel File Systems

Monterna nu resten av kernelns filsystem:

```
mount -vt devpts devpts $LFS/dev/pts -o gid=5,mode=620
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
mount -vt tmpfs tmpfs $LFS/run
```

Vad alternativen betyder:

gid=5

Säkerställer att alla devpts-created enhetsnoder ägs av gruppen med ID 5. Detta är det ID vilket vi senare kommer att använda för gruppen *tty*. Vi använder ett grupp-ID istället för ett namn, eftersom värdsystemet kanske använder ett annat ID för sin tty-grupp.

mode=0620

Säkerställer att alla devpts-created enhetsnoder är satta till mode 0620(skrivbar och läsbar för användare, skrivbar för grupp). Tillsammans med alternativet ovan, säkerställer detta att devpts skapar enhetsnoder vilka lever upp till kraven ställda av grantpt(), vilket innebär att Glibc’s **pt_chown** binär (som ej installeras som standard) inte behövs.

I vissa värdsystem är /dev /shm en symbolisk länk till /run /shm. Tmpfs för /run monterades redan ovan, så i detta fall är det endast nödvändigt att skapa en mapp:

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

6.3. Pakethantering

Pakethantering är ett ämne som folk ofta önskar att vi tar upp i denna bok. En pakethanterare möjliggör spårning av installerade paket, vilket gör det enkelt att ta bort samt uppgradera dem. Utöver hanteringen av binärer och bibliotek, sköter pakethanterare även om installationen av konfigurationsfiler. Innan du börjar undra, så NEJ – denna sektion kommer inte diskutera eller rekommendera någon särskild pakethanterare. Vad den tillhandahåller är en övergripande beskrivning vad gäller de mer populära teknikerna, samt hur de fungerar. Den pakethanterar som är perfekt för dig, kan vara en av dessa tekniker, eller en kombination av flera av dem. I övrigt kan problem uppstå vid uppgradering av paket.

Ett antal anledningar till varför ingen pakethanterare nämns i LFS och BLFS, är:

- Att diskutera pakethantering leder bort fokus från det som är tänkt att vara dessa böckers mål – nämnligen hur ett Linux-system byggs.
- Det finns flera alternativ för pakethantering, vart och ett med svagheter och styrkor. Att inkludera ett alternativ som skulle tillfredställa alla målgrupper vore svårt.

Vi har ändock inkluderat ett antal tips vad gäller pakethantering. Besök *Hints Project* och se om du finner något vilket är av intresse.

6.3.1. Problem med uppgraderingar

En pakethanterare gör det enklare att uppgradera paket till nyare versioner. Instruktionerna i LFS och BLFS kan rent generellt sett användas för att uppgradera till nyare versioner. Här är ett antal aspekter vilka du bör vara medveten om ifall du ska uppgradera paket, särskilt om du gör det på ett system vilket är aktivt:

- Om Glibc behöver uppgraderas till en nyare version (t.ex från glibc-2.19 till glibc-2.20), så är det säkrare att bygga om hela LFS. Även ifall det kan gå att bygga om alla paket i deras specifierade ordning, rekommenderar vi det inte.
- Ifall ett paket som innehåller ett delat bibliotek förändras, då måste alla paket vilka är dynamiskt länkade till biblioteket kompileras om för att länka till det nya biblioteket. (Notera att det inte finns någon sorts korrelation mellan Paketversionen och bibliotekets namn). Föreställ dig t.ex ett paket foo-1.2.3 som installerar ett delat paket vid namn Libfoo.so.1. Säg att du uppgraderar till ett nyare paket foo-1.2.4 vilket installerar ett paket libfoo.so.2. i detta fall måste alla paket som är dynamiskt länkade till libfoo.so.1 kompileras om så de länkar till libfoo.so.2. Notera att du inte bör ta bort tidigare bibliotek innan det att de beroende paketen har kompilerats om.

6.3.2. Tekniker för pakethantering

Följande representerar ett antal vanliga tekniker för pakethantering. Innan du väljer en specifik pakethanterare, läs upp dig angående deras olika funktioner, styrkor och särskilt svagheter.

6.3.2.1. Jag har allt i mitt huvud!

Ja, detta är en pakethanteringsteknik. Vissa behöver ingen pakethanterare, för de har en väldigt god kännedom om de paket vilka de behöver, och vilka filer som är installerade av varje sådant paket. Vissa användare behöver inte heller en pakethanterare, för de planerar att bygga om hela systemet då ett paket ändras.

6.3.2.2. Installering i separata mappar

Detta är en enkelt variant av pakethantering, vilken inte kräver ett separat paket för att hantera installationerna. Varje paket installeras i en separat mapp. Exempelvis foo-1.1 installeras i /usr /pkg /foo-1.1 och en symbolisk länk som leder från /usr /pkg /foo till /usr /pkg /foo-1.1. När en ny version av foo installeras, installeras den i /usr /pkg /foo-1.2 och den tidigare symlinken ersätts av en sådan vilken leder till den nya versionen av foo.

Miljövariabler såsom PATH, LD_LIBRARY_PATH, MANPATH, INFOPATH och CPPFLAGS behöver expanderas så att de inkluderar /usr/pkg/foo. Denna typ av pakethantering blir ohållbar då fler än ett mindre antal paket installeras.

6.3.2.3. Paket-hantering med hjälp av Symlink

Detta är en variation på den föregående tekniken för pakethantering. Varje paket installeras på ett sätt likt i den tidigare tekniken, men istället för att skapa symlänken, länkas varje fil symboliskt in i /usr hierarkin. Detta innebär att man inte behöver expandera miljövariablerna. Även ifall symlänkarna kan skapas av användaren för att automatisera skapandet, har många pakethanterare skapats vilka använder denna metodik. Till dessa räknas Stow, Epkg, Graft och Depot.

Installationen behöver fakeas, så att paketet tror att det är installerat i /usr när det egentligen installerats i hierarkin /usr/pkg. Detta är inte alltid en enkel uppgift att lyckas med. Föreställ dig exempelvis att du installerar paketet libfoo-1.1. Följande instruktioner kanske inte installerar paketet på ett korrekt sätt:

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

Installationen kommer att fungera, men de beroende paketen kanske inte kommer länka till libfoo på det sätt vilket du förväntar dig. Om du kompilerar ett paket som länkar till libfoo, kommer du kanske märka att det är länkat till /usr/pkg/libfoo/1.1/lib/libfoo.so.1 istället för som förväntat till /usr /lib /libfoo.so.1. Den korrekta metodiken är att använda DESTDIR för att imitera installationen av paketet. Detta görs enligt följande:

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

De flesta paket stödjer denna metodik, men det finns de som inte gör det. För de paket som inte tillhandahåller stöd, kan du behöva manuellt installera paketet. Det kan även vara enklare att installera problematiska paket i /opt. .

6.3.2.4. Tidsstämpel baserade

Med denna teknik så tidsstämplas ett paket innan att det installeras. Efter installationen kan en vanlig användning av kommandot **find** generera en log med med alla filer vilka installerats efter att tidsstämpeln skapades. En pakethanterare aom använder denna teknik är install-log.

Även ifall denna metodik har fördelen att vara väldigt rättfram, har den två nackdelar. Om under installation filerna installeras med någon annan tidsstämpel än den nuvarande tiden, kommer dessa filer inte att spåras av pakethanteraren. Denna metodik kan också endast användas då ett paket i taget installeras. Loggarna vilka genereras är inte tillförlitliga om två paket installeras på två separata konsoler.

6.3.2.5. Spårande av installations-skript

Med denna metodik sparas kommandona som installations-skripten utför. Det finns två tekniker vilka kan användas:

Miljövariabeln LD_PRELOAD kan definieras för att peka till ett bibliotek vilket förladdas innan installation. Under själva installationen spårar detta bibliotek de paket vilka installeras, genom att bifoga sig självt till diverse körbara filer likt **cp, install, mv** samtidigt som det även spårar systemanrop vilka modifierar filsystemet. För att denna metodik ska fungera, måste alla körbara filer vara dynamiskt länkade utan suid eller sgid biten. Att förladda biblioteket kan leda till oönskade sidoeffekter vid installation. Det är rekommenderas därför att man genomför ett antal tester, så det går att säkerställa att pakethanteraren inte förstör något samt loggar alla korrekta filer.

Nästa teknik är **strace**, vilken loggar alla systemanrop vilka utförs då installations-skriptet exekveras.

6.3.2.6. Att skapa paket-arkiv

Med denna metodik så fakeas paketinstallationen in i ett separat träd, på ett sätt vilket beskrivs i metodiken för hantering av paket med hjälp av symlänkar. Efter installationen skapas ett paketarkiv med hjälp av de installerade filerna. Detta arkiv används sedan för att installera paketet antingen på den lokala maskinen eller på andra maskiner.

Denna metodik används av de flesta pakethanterare vilka återfinns i kommersiella distributioner. Detta inkluderar t.ex RPM (vilken krävs av *Linux Standard Base Specification*), pkg-utils, Debian's apt, och Gentoo's Portage. Information om hur den här typen av pakethantering kan implementeras i LFS återfinns här: *http://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt*.

Skapandet av paket vilka inkluderar beroenden är komplext, och inte inkluderat i LFS.

Slackware använder ett tar-baserat system för paketarkiv. Detta system hanterar inte paketberoenden på samma sätt som mer komplexa pakethanterare gör. För detaljer angående hur Slackware fungerar, läs: *http://www.slackbook.org/html/package-management.html.*

6.3.2.7. Användarbaserad administration

Denna metodik, unik för LFS, skapades av Matthias Benkmann, och finns tillgänlig i *Hints Project*. Med denna metodik installeras varje paket, i standard-mapparna, som en separat användare. Filer vilka tillhör ett paket är enkelt identifierade genom sitt user-id. Funktionaliteten samt nackdelarna med denna metodik är för komplicerade för att diskuteras i denna sektion. För mer info, läs: *http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt.*

6.3.3. Att driftsätta LFS på multipla system

En av fördelarna vilket ett LFS-system har, är att det inte finns några filer vilka är beroende av filers positioner på systemet. Att klona ett LFS-system till en annan dator med samma arkitektur som ursprungssystemet, kan göras genom att använda **tar** på den LFS-partition vilken innehåller root-mappen (omkring 250MB okomprimerat för LFS), kopiera denna fil till den nya systemet via nätverksdelning eller USB, och sedan extrahera denna fil. Ett antal konfigurationsfiler måste därefter justeras. Dessa inkluderar bland annat: /etc/hosts, /etc/fstab, /etc/passwd, /etc/group, /etc/shadow, /etc/ld.so.conf, /etc/sysconfig/rc.site, /etc/sysconfig/network, and /etc/sysconfig/ifconfig.eth0.

En kernel vilken tar hänsyn till skillnader emellan de olika systemens underliggade hårdvara kan behöva byggas.

Kommentar

Det har kommit in rapporter om problem då man kopierat emellan snarlika men inte identiska arkitekturer. Exempelvis instruktionerna i ett Intel-system är inte identiska med de i en AMD-processor, samtidigt som senare versioner av vissa processorer har instruktioner vilka inte finns tillgängliga i tidigare versioner.

Systemet måste slutligen göras uppstartbart, men detta återkommer vi till i sektion 8.4.

6.4. Träda in i Chroot-miljön

Det är dags att träda in i chroot-miljön, för att börja bygga och installera det slutgiltiga LFS-systemet. Som användare root, kör nedan kommando för att träda in i miljön som tillsvidare endast är befolkad med de temporära verktygen:

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

Tillvalet -i vilket ges till kommandot **env** kommer rensa chroot-miljöns alla variabler. Efter detta definieras endast variablerna för HOME, TERM, PS1 och PATH. Construct TERM=$TERM kommer definiera variabeln TERM som existerar inom chroot till samma värde som utanför chroot. Denna variabel krävs för att paket som Less och Vim ska fungera korrekt. Om andra variabler krävs, som CFLAGS eller CXXFLAGS, är nu en bra tidpunkt att definiera dem.

Från denna punkt framåt finns det ingen mening med att variabeln LFS längre, eftersom alla händelser kommer att begränsas till filsystemet LFS. Detta beror på att Bash-skalet instrueras att $LFS nu är root (/) mappen.

Uppmärksamma att /tools /bin listas sist i PATH. Detta innebär att ett temporärt verktyg inte längre kommer användas då dess slutgiltiga version installerats. Detta inträffar då skalet inte “kommer ihåg” platsen för exekverade binärer. Det är på grund av detta som vi stängde av hashing med Bash, genom att skicka med flaggan -h.

Notera att Bash kommer returnera “i have no name!”. Detta är normalt, eftersom filen /etc /passwd inte skapats ännu.

Kommentar

Det är viktigt att alla kommandon i nästa och efterföljande kapitel körs från chroot-miljön. Om du av någon anledning lämnar denna miljö(genom att exempelvis starta om datorn), säkerställ att de virtuella kernel filsystemen är monterade som beskrivet i sektion 6.2.2 och 6.2.3 och träd in i chroot igen innan det att du fortsätter med kompileringen och installationen.

6.5. Att skapa mappar

Det är nu dags att infoga lite struktur i LFS-systemet. Skapa ett standard mapp-träd genom att använda följande kommando:

```
mkdir -pv /{bin,boot,etc/{opt,sysconfig},home,lib/firmware,mnt,opt}
mkdir -pv /{media/{floppy,cdrom},sbin,srv,var}
install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{color,dict,doc,info,locale,man}
mkdir -v  /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -v  /usr/libexec
mkdir -pv /usr/{,local/}share/man/man{1..8}
mkdir -v  /usr/lib/pkgconfig

case $(uname -m) in
 x86_64) mkdir -v /lib64 ;;
esac

mkdir -v /var/{log,mail,spool}
ln -sv /run /var/run
ln -sv /run/lock /var/lock
mkdir -pv /var/{opt,cache,lib/{color,misc,locate},local}
```

Mappar är, som standard, skapade med rättigheter satta till 755, men detta är inte önskvärt för alla mappar. Genom kommandona ovan sker två förändringar – en till användaren root's /home, och en till mappar för de temporära filerna.

Den första ändringen säkerställer att inte vem som helst kan ta sig in in /root – samma ändring som en vanlig användare skulle göra med sin egen /home. Den andra ändringen säkerställer att alla användare kan skriva till /tmp och /var/tmp, men däremot inte kan ta bort en annan användares filer från dem. Det senare är förhindrat med hjälp av den "klibbiga" biten. Denna bit är den högsta biten (1) i rättighetsinställningen 1777.

6.5.1. FHS samtyckes kommentar

Mappträdet är baserat på Filesystem Hierarchy Standard (FHS) tillgänglig via https://refspecs.linuxfoundation.org/fhs. *Shtml).* FHS specifierar även tillvalet av valfria mappar likt /usr /local /games och /usr /share /games. Vi skapar däremot endast de mappar vilka vi behöver. Om du vill skapa dessa alternativa mappar så är det ok.

6.6. Skapa nödvändiga filer och symlänkar

Vissa program använder hårdkodade sökvägar till program vilka inte existerar ännu. För att tillfredställa dessa program, skapa ett antal symboliska länkar vilka kommer ersättas av riktiga filer allteftersom detta kapitel fortgår.

```
ln -sv /tools/bin/{bash,cat,chmod,dd,echo,ln,mkdir,pwd,rm,stty,touch} /bin
ln -sv /tools/bin/{env,install,perl,printf}         /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1}                  /usr/lib
ln -sv /tools/lib/libstdc++.{a,so{,.6}}             /usr/lib

ln -sv bash /bin/sh
```

Syftet bakom varje länk:

/bin/bash
Många bash-skript specifierar`/bin/bash`.

/bin/cat
Detta sökvägs-namn är hårdkodat in i Glibc’s configure-skript.

/bin/dd
Sökvägen till **dd** kommer hårdkodas in i /usr /bin /libtool verktyget.

/bin/echo
För att tillfredställa ett av testen i Glibc’s test-svit, vilken förväntar sig`/bin/echo`.

/usr/bin/env
Detta sökvägs-namn är hårdkodat in i vissa pakets bygg-procedurer.

/usr/bin/install
Sökvägen till install kommer hårdkodas in i filen /usr /lib /bash /Makefile.inc

/bin/ln
Sökvägen till ln kommer hårdkodas in i filen /usr/lib/perl5/5.30.1/<target-triplet>/Config_heavy.pl

/bin/pwd
Vissa **configure**-skript, särskilt glibc’s, har detta sökvägs-namn hårdkodat.

/bin/rm
Sökvägen till **rm** kommer att hårdkodas in i filen /usr/lib/perl5/5.30.1/<target-triplet>/Config_heavy.pl

/bin/stty
Detta sökvägs-namn är hårdkodat in i Expect. Det är därför nödvändigt för test-sviter till Binutils och GCC.

/usr/bin/perl
Många Perl-skript hårdkodar denna sökväg till programmet Perl.

/usr/lib/libgcc_s.so{,.1}
Glibc behöver denna för att biblioteket pthreads ska fungera.

/usr/lib/libstdc++{,.6}
Behövs av flertalet av testen i Glibc’s test-svit, och för C++ support i GMP.

/bin/sh
Många skal-skript hårdkodar `/bin/sh`.

Historiskt sett så lagrar Linux en lista av alla monterade filsystem, i filen /etc /mtab. Moderna kernels underhåller denna lista internt, och användare kan få tillgång till den via filsystemet /proc. För att tillfredställa verktyg vilka förväntar sig närvaron av /etc /mtab, skapa följande symboliska länk:

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

Så att det är möjligt för root att logga in - och för att namnet root skall erkännas - måste det finnas relevanta sektioner i filerna /etc /passwd och /etc /group.

Skapa filen `/etc/passwd` Genom att köra följande kommando:

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/bin/false
daemon:x:6:6:Daemon User:/dev/null:/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/var/run/dbus:/bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
EOF
```

Det verkligen lösenordet för root (det “x” som används här är endast temporärt) kommer att sättas senare.

Skapa filen /etc /group genom att köra följande 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:
wheel:x:97:
nogroup:x:99:
users:x:999:
EOF
```

De skapade grupperna är inte del av någon standard – de är skapade dels till följd av de krav vilka ställs av Udev's konfiguration i detta kapitel, dels till följd av konventioner vilka används av ett antal existerande Linux-distributioner. Utöver detta förlitar sig vissa test-sviter på specifika användare och grupper. Linux Standard Base(LSB) rekommenderar endast att, bortsett gruppen "root" med ett grupp-ID(GID) av 0, en grupp "bin" med ett GID av 1 finns tillgänglig. Alla andra gruppnamn och GID's kan väljas fritt, då välskrivna program inte beror på GID-nummer utan istället gruppnamn.

För att få bort svaret "i have no name!", starta ett nytt skal. Eftersom ett komplett Glibc installerades i kapitel 5, och även /etc /passwd och /etc /group har skapats, kommer användarnamn och gruppnamn resolution nu fungera:

```
exec /tools/bin/bash --login +h
```

Notera användningen av direktivet +h. Detta specifierar för Bash att inte använda sitt interna sökvägs-hash. Utan detta direktiv skulle bash komma ihåg sökvägarna till de binärer vilka det har exekverat. För att säkerställa användandet av de nykompilerade binärer så snart de installerats, kommer +h används genom hela kapitlet.

Programmen **login**, **agetty**, och **init** (samt andra) använder ett antal logg-filer för att spara information såsom vem som har loggat in i systemet och när. Dessa program kommer dock inte skriva till logg-filerna ifall dessa inte redan existerar. initialisera loggfilerna och ge dem korrekt behörigheter:

```
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
```

Filen /var /log /wtmp sparar all inloggningar och utloggningar. Filen /var /log /lastlog sparar när varje användare senast loggade in. Filen /var /log /faillog sparar misslyckade inloggningsförsök. Filen /var /log /btmp sparar inkorrekta försök att logga in.

Kommentar

Filen`/run/utmp` sparar vilka användare som för tillfället är inloggade. Denna fil skapas dynamiskt i själva boot-skripten.

6.7. Linux-5.5.3 API Headers

Paketet Linux API Headers (i linux-5.5.3.tar.xz) öppnar upp kerneln's API för användning av Glibc.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 1 GB

6.7.1. Installation av Linux API Headers

Linux kernel behöver öppna upp ett Application Programming Interface(API) vilket systemets C bibliotek kan använda. detta görs genom att sanitisera diverse C headerfiler som medföljer Linux kernelns tarball.
Säkerställ att det inte finns några stale files inbäddade i paketet:

```
make mrproper
```

Extrahera nu de användarsynliga kernelheadersen från source. Det rekommenderade make-target “headers_install” kan inte användas, eftersom det kräver rsync, vilket inte finns tillgänligt i /tools. Headersen placeras först i ./usr, sedan tas ett antal filer vilka används av kernel utvecklarna bort, och sedan kopieras filerna till sin slutgiltiga plats.

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

6.7.2. Innehåll Linux API Headers

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

Korta beskrivningar

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

6.8. Man-pages-5.05

Paketet Man-pages innehåller mer än 2,200 man sidor.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 31 MB

6.8.1. Installation av Man-pages

Installera man genom att köra:

```
make install
```

6.8.2. Innehåll Man-pages

Installerade filer: various man pages

Korta beskrivningar

`man pages` Förklaringar vad gäller program, viktiga enhets-filer, och signifikanta konfigurations-filer.

6.9. Glibc-2.31

Paketet Glibc innehåller det huvudsakliga C-biblioteket. Detta bibliotek tillhandahåller de grundläggande rutinerna för att allokera minne, söka igenom mappar, öppna och stänga filer, läsa och skriva filer, stränghantering, och så vidare.

Förväntad byggtid: 19 SBU
Nödvändigt disk-utrymme: 5.5 GB

6.9.1. Installation av Glibc

Kommentar

Byggsystemet för Glibc är självtillräckligt och kommer installeras perfekt, även fast kompilatorns spec-filer och länkare fortfarande pekar till /tools. Specs och länkare kan inte justeras förräns att Glibc installerats, då Glibc autoconf test skulle ge falska resultat och motverka syftet att skapa en ren byggnation.

Vissa av Glibc's program använder mappen /var /db, som inte lever upp till FHS krav, för att lagra runtime data. Applicera följande patch för att säkerställa att sådana program lagrar sin data i platser vilka lever upp till FHS krav.

```
patch -Np1 -i ../glibc-2.31-fhs-1.patch
```

Skapa en symlink enligt LSB-kraven. Ytterligare, för x86_64, skapa en symlink som krävs för att den dynamiska laddaren ska fungera korrekt:

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

Glibc dokumentationen rekommenderar att Glibc byggs i en dedikerad bygg-mapp:

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

Förbered för kompilering:

```
CC="gcc -ffile-prefix-map=/tools=/usr" \
../configure --prefix=/usr                      \
             --disable-werror                   \
             --enable-kernel=3.2                \
             --enable-stack-protector=strong    \
             --with-headers=/usr/include        \
             libc_cv_slibdir=/lib
```

Vad alternativen betyder:

`CC="gcc -ffile-prefix-map=/tools=/usr"`
Se till att GCC lagrar alla referenser till filer i /tools, vilka är resultat av kompileringen, som ifall dessa filer återfinns i /usr. Detta undviker att inkorrekta sökvägar introduceras i felsöknings-symbolerna.

`--disable-werror`
Avaktiverar alternativet -Werror skickat till GCC. Detta är nödvändigt för att köra testsviten.

`--enable-stack-protector=strong`
Ökar systemets säkerhet, genom att lägga till extra kod vilken söker efter buffer overflows, såsom *stack smashing* attacker.

`--with-headers=/usr/include`
Specifierar för byggsystemet var det kan hitta kernelns API headers. Som standard eftersöks dessa headers i /tools /include.

`libc_cv_slibdir=/lib`
Denna variabel definierar det korrekta biblioteket för alla system. Vi vill inte att lib64 används.

Kompilera paketet:

```
make
```

Viktigt

i denna sektion betraktas testsviten för Glibc som kritisk. Skippa den inte under några omständigheter!

Vanligtvis misslyckas ett antal av testen. Misslyckanden som listas nedan är vanligtvis ok att ignorera.

```
case $(uname -m) in
  i?86)   ln -sfnv $PWD/elf/ld-linux.so.2        /lib ;;
  x86_64) ln -sfnv $PWD/elf/ld-linux-x86-64.so.2 /lib ;;
esac
```

Kommentar

Den symboliska länken ovan behövs för att det i detta stadie ska gå att köra testen i **chroot** miljön. Den kommer att skrivas över under installerings-fasen nedan.

```
make check
```

Du kommer att se ett antal testmisslyckanden rapporteras. Testsviten för Glibc är relativt beroende av värdsystemet. Detta är en lista av de vanligaste rapporterade problemen för vissa versioner av LFS:

- *misc/tst-ttyname* fallerar ofta i LFS's chroot miljö
- *inet/tst-idna_name_classify* fallerar ofta i LFS chroot miljö.
- *posix/tst-getaddrinfo4* och *posix/tst-getaddrinfo5* kan fallera på vissa arkitekturer.
- *nss/tst-nss-files-hosts-multi* testet kan fallerar till följd av orsaker som ännu inte är kända.
- *rt/tst-cputimer{1,2,3}* testen är beroende av värdens kernel. Bland annat kernels 4.14.91–4.14.96, 4.19.13–4.19.18, Och 4.20.0-4.20.5 kan leda till att dessa test fallerar.
- Math-testen fallerar ibland när de körs på ett system där CPUn inte är en relativt ny Intel eller AMD processor.

Även ifall det är en harmlös varning, så kommer Glibc's installations stadie att klaga om att `/etc/ld.so.conf`. inte kan upptäckas. Lös detta med hjälp av:

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

Fixa den genererade make-filen för att skippa en onödig kontroll vilken fallerar i LFS's partiellt byggda tillstånd:

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

Installera paketet:

```
make install
```

Installera konfigurationsfilen och runtime-mappen för **nscd**:

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

Installera därefter de locales som kan få systemet att svara på ett annat språk. Inga av dessa locales är nödvändiga, men ifall vissa av dem saknas kommer vissa av de framtida testsviterna att hoppa över viktiga test.

Enskilda locales kan installeras med hjälp av programmet localedef. Det första localedef-kommandot nedan kombinerar /usr /share /i18n /locales /cs_CZ charset-självständiga locale definition med /urs /share /i18n /charmaps /UTF-8.gz charmap definition, och appendar resultatet till filen /usr /lib /locale /locale-archive. Nedan instruktioner kommer att installera de locales vilka krävs för att bidra med optimalt stöd till testsviterna:

```
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 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_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 it_IT -f ISO-8859-1 it_IT
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.SIJS 2> /dev/null || true
localedef -i ja_JP -f UTF-8 ja_JP.UTF-8
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.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
```

Installera även localen för för ditt eget land, språk och karaktärs-set.

Alternativt, installera alla locales listade i `glibc-2.31/localedata/SUPPORTED` Filen (Den inkluderar alla locales listade ovan och många fler) med hjälp av följande tidskrävande kommando:

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

Använd sedan kommandot **localedef** för att skapa och installera locales ej listade i filen glibc-2.31/localedata/SUPPORTED. Du kommer antagligen inte att behöva dem dock.

Kommentar

Glibc använder nu libidn2 när den resolving internationaliserade domännamn. Detta är ett run time beroende. Om denna förmåga krävs, då återfinns instruktioner för installation av libidn2 i *BLFS libidn2 page.*

6.9.2. Konfigurering av Glibc

6.9.2.1. Lägg till nsswitch.conf

Filen /etc/nsswitch.conf måste skapas eftersom Glibc's standarder inte fungerar så bra i en nätverkad miljö.

Skapa en ny fil `/etc/nsswitch.conf` genom att köra följande:

```
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
```

6.9.2.2. Lägga till data för tidszon

Installera och specifiera data för tidszon genom att använda följande kommando:

```
tar -xf ../../tzdata2019c.tar.gz

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

for tz in etcetera southamerica northamerica europe africa antarctica  \
          asia australasia backward pacificnew systemv; 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
```

Vad alternativen betyder:

`zic -L /dev/null ...`
Skapar posix tidszoner, utan leap seconds. Konventionellt sett så placeras dessa i både filen zoneinfo samt filen zoneinfo/posix. Det är nödvändigt att placera POSIX tidszonerna i zoneinfo, annars kommer diverse testsviter att rapportera errors. På ett inbäddat system, där utrymme är minimalt och du inte planerar att uppdatera tidszoner, kan du antagligen spara 1.9MB genom att inte använda mappen posix, men då kommer vissa applikationer och test kanske att producera errors.

`zic -L leapseconds ...`
Skapar korrekta tidszoner, inkluderat leap seconds. På ett inbäddat system där utrymme är minimalt och du inte planerar att uppdatera tidszoner, kan du antagligen spara 1.9MB genom att inte använda mappen right.

`zic ... -p ...`
Skapar filen posixrules.

För att avgöra den lokala tidszonen kan man köra följande skript:

```
tzselect
```

Efter att man svarat på ett antal frågor angående platsen, kommer skriptet outputa namnet på tidszonen (t.ex America/Edmonton). Det finns också andra möjliga tidszoner listade i /usr /share /zoneinfo, vilka inte identifieras av skriptet men som oavsett detta kan användas.

Skapa sedan filen `/etc/localtime` genom att köra:

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

Ersätt *`<xxx>`* med namnet på den valda zonen.

6.9.2.3. Att konfigurera den dynamiska laddaren

Som standard kommer den dynamiska laddaren (/lib /ld-linux.so.2) söka igenom /lib och /usr /lib för att hitta dynamiska bibliotek vilka behövs av program medans det att de körs. Dock, ifall det finns paket i andra mappar än /lib och /usr/lib måste dessa läggas till i filen /etc/ld.so.conf för att den dynamiska laddaren ska kunna finna dem. Två mappar som vanligtvis innehåller extra bibliotek är /usr/local/lib och /opt/lib, så lägg till dessa mappar till den dynamiska laddarens sökvägar.

Skapa den nya filen `/etc/ld.so.conf` genom att köra följande:

```
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf
/usr/local/lib
/opt/lib

EOF
```

Om så önskas, kan den dynamiska laddaren även söka igenom en mapp och inkludera innehållet av filerna vilka hittas där. Filerna i denna include-mapp innehåller vanligtvis en rad, vilken specifierar den önskade biblioteks-sökvägen. för att tillföra denna funktionalitet, kör följande kommando:

```
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
```

6.9.3. Innehåll Glibc

Installerade program: catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, makedb, mtrace, nscd, pcprofiledump, pldd, sln, sotruss, sprof, tzselect, xtrace, zdump, and zic

Installerade bibliotek: ld-2.31.so, libBrokenLocale.{a,so}, libSegFault.so, libanl.{a,so}, libc.{a,so}, libc_nonshared.a, libcrypt.{a,so}, libdl.{a,so}, libg.a, libm.{a,so}, libmcheck.a, libmemusage.so, libmvec.{a,so}, libnsl.{a,so}, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libpcprofile.so, libpthread.{a,so}, libpthread_nonshared.a, libresolv.{a,so}, librt.{a,so}, libthread_db.so, and libutil.{a,so}

Installerade mappar: /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, and /var/lib/nss_db

Korta beskrivningar

catchsegv Kan användas för att skapa ett stack trace då ett program avslutas pga av ett segmentation fault

gencat Genererar meddelande-kataloger

getconf Visar systemkonfigurationsvärden för filsystems-specifika variabler

getent	Mottar entries från en administrativ databas
iconv	Genomför character set omvandling
iconvconfig	Skapar snabbladdande **iconv** module konfigurations filer
ldconfig	Konfigurerar den dynamiska länkarens runtime bindings
ldd	Rapporterar vilka delade bibliotek som som krävs av varje givet program eller delat bibilotek
lddlibc4	Assisterar **ldd** med objektfiler
locale	Printar diverse information om nuvarande locale
localedef	Kompilerar locale specifikationer
makedb	Skapar en simpel databas med hjälp av textinput
mtrace	Läser och tolkar en memory trace fil, och visar en sammanställning i människo-läsbart format
nscd	En daemon som tillhandahåller en cache för de vanligaste name-service förfrågningarna
pcprofiledump	Dumpar information genererad av PC profilering
pldd	Listar dynamiska delade objekt vilka används av aktiva processer
sln	Ett statiskt länkat **ln** program
sotruss	Spårar delade biblioteks procedur-anrop som är relaterade till ett specifikt kommando
sprof	Läser och visar profilerings-data för delade objekt
tzselect	Frågar användaren om systemets geografiska plats och rapporterar den motsvarande tidszonen
xtrace	Spårar exekveringen av ett program, genom att skriva den nuvarande exekverande funktionen
zdump	Tidszons dumpare
zic	Tidszons kompilator
`ld-2.31.so`	Hjälpprogrammet för delade biblioteks executables
`libBrokenLocale`	Används internt av Glibc som ett hack för att få trasiga program (t.ex vissa Motif applikationer) att exekvera. Se kommentarer i glibc-2.31/locale/broken_cur_max.c för mer information
`libSegFault`	Signalhanteraren för segmentationsfel, används av **catchsegv**
`libanl`	Ett asynkront bibliotek för namn lookup
`libc`	Det huvudsakliga C biblioteket
`libcrypt`	Det kryptografiska biblioteket
`libdl`	Det dynamiskt länkande gränssnitts biblioteket
`libg`	Låtsasobjekts bibliotek som inte innehåller riktiga funktioner. Tidigare runtime library for **g++**
`libm`	Det matematiska biblioteket
`libmcheck`	Aktiverar minnesallokering när länkat till
`libmemusage`	Används av **memusage** för att hjälpa samla information om ett programs minnes-användning
`libnsl`	Biblioteket för nätverks-tjänster
`libnss`	Name Service Switch bibliotek, innehåller funktioner för att resolving värdnamn, användarnamn, Gruppnamn, alias, tjänster, protokoll, etc.

`libpcprofile`	Can be preloaded to PC profile an executable
`libpthread`	Biblioteket för POSIX trådar
`libresolv`	Innerhåller funktioner för att skapa, skicka, och tolka paket, i relation till Internet domän-servar
`librt`	Innehåller funktioner vilka tillhandahåller de flesta gränssnitt vilka specifieras i POSIX.1b realtids extensions
`libthread_db`	Innehåller funktioner användbara för att bygga felsökare för mång-trådade program
`libutil`	Innehåller kod för “standard”-funktioner vilka används i många olika Unix-verktyg

6.10. Att justera Toolchainen

Då de slutgiltiga C-biblioteken nu har installerats, är det dags att justera toolchainen så att den kommer länka alla paket som i framtiden kompileras emot dessa nya bibliotek.

Säkerhetskopiera först länkaren i /tools, och ersätt den med den justerade länkare vilken vi skapade i kapitel 5. Vi kommer även skapa en länk till dess motpart i /tools /$(uname -m) -pc-linux-gnu/bin:

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(uname -m)-pc-linux-gnu/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(uname -m)-pc-linux-gnu/bin/ld
```

Anpassa därefter GCCs specifikationsfil, så att den pekar gentemot den nya dynamiska länkaren. Att helt enkelt ta bort alla instanser av “/tools” borde leda till att den korrekta pathen till den dynamiska länkaren återstår. Justera även specifikationsfilen så att GCC vet var korrekta headers och GLIBC filer kan återfinnas. Ett **sed** kommando löser detta:

```
gcc -dumpspecs | sed -e 's@/tools@@g'                   \
    -e '/\*startfile_prefix_spec:/{n;s@.*@/usr/lib/ @}' \
    -e '/\*cpp:/{n;s@$@ -isystem /usr/include@}' >      \
    `dirname $(gcc --print-libgcc-file-name)`/specs
```

Det är en bra idé att visuellt säkerställa att de önskade förändringarna faktiskt genomfördes.

Det är helt nödvändigt att vid detta stadie säkerställa att alla grundläggande funktioner (kompilering och länkning) hos den anpassade nya toolchainen fungerar som förväntat. För att kolla detta, kör följande integritets-test:

```
echo 'int main(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

Inga errors borde rapporteras, och det sista kommandots output kommer vara (undantaget plattformsspecifika skillnader i den dynamiska länkarens namn):

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

Notera att på 64-bit systems är`/lib` Platsen för vår dynamiska länkare, men den tillträds via en symbolisk länk i /lib64.

Kommentar

På 32bits system bör tolken vara /lib/ld-linux.so.2.

Säkerställ nu att vi är redo att använda de korrekta uppstartsfilerna:

```
grep -o '/usr/lib.*/crt[1in].*succeeded' dummy.log
```

Det sista kommandots output bör vara:

```
/usr/lib/../lib/crt1.o succeeded
/usr/lib/../lib/crti.o succeeded
/usr/lib/../lib/crtn.o succeeded
```

Verifiera att kompilatorn söker efter de korrekta headerfilerna:

```
grep -B1 '^ /usr/include' dummy.log
```

Detta kommando bör returnera nedan output:

```
#include <...> search starts here:
 /usr/include
```

Verifiera sedan att den nya länkaren använder de korrekta sökvägarna:

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

Referenser till sökvägar vilka har komponenter med '-linux-gnu' bör ignoreras, men bortsett från detta bör det sista kommandots output vara:

```
SEARCH_DIR("/usr/lib")
SEARCH_DIR("/lib")
```

Säkerställ sedan att vi använder det korrekta libc:

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

Det sista kommandots output borde vara:

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

Slutligen, säkerställ att GCC använder den korrekta dynamiska länkaren:

```
grep found dummy.log
```

Det sista kommandots output borde vara (undantaget plattformsspecifika skillnader i den dynamiska länkarens namn):

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

Om den output som visas inte motsvarar vad som kan ses ovan, är något allvarligt fel. Undersök och felsök varje steg i processen, tills du hittar vad felet är. Mest sannolikt är att något gick fel då specifikationsfilen modifierades. Problem i detta steg måste lösas innan det att du fortsätter vidare.

När allting väl fungerar som det är tänkt, städa upp testfilerna:

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

6.11. Zlib-1.2.11

Paketet Zlib innehåller rutiner för komprimering och dekomprimering.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 5.1 MB

6.11.1. Installation of Zlib

Förbered Zlib för kompilation:

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

Kompilera:

```
make
```

Testa:

```
make check
```

Installera:

```
make install
```

Det delade biblioteket behöver flyttas till /lib, och till följd av detta behöver .so filen i */usr/lib* återskapas:

```
mv -v /usr/lib/libz.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libz.so) /usr/lib/libz.so
```

6.11.2. Innehåll Zlib

Installerade bibliotek: libz.{a,so}

Korta beskrivningar

`libz` Paketet Zlib innehåller rutiner för komprimering och dekomprimering.

6.12. Bzip2-1.0.8

Paketet Bzip2 innehåller program för komprimering och dekomprimering av filer. Att komprimera textfiler med Bzip2 istället för med hjälp av det traditionella Gzip, innebär en mycket högre procentuell grad av komprimering.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 7.7 MB

6.12.1. Installation av Bzip2

Applicera en patch vilken kommer installera dokumentationen för detta paket:

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

Följande kommando säkerställer att installationer av symboliska länkar är relativa:

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

Säkerställer att man pages installeras i den korrekta platsen:

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

Förbered för kompilering:

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

Vad parametrarna betyder:

`-f Makefile-libbz2_so`
Detta leder till att Bzip2 byggs med hjälp av en annan make-fil, i detta fall Makefile-libbz2_so. Denna skapar ett dynamiskt libbz2.so bibliotek och länkar Bzip2 verktygen gentemot det.

Kompilera och testa paketet:

```
make
```

Installera:

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

Installera den delade **bzip2** binären i mappen /bin. Skapa sedan de nödvändiga symboliska länkarna, och städa sedan upp:

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

6.12.2. Innehåll Bzip2

Installerade program: bunzip2 (link to bzip2), bzcat (link to bzip2), bzcmp (link to bzdiff), bzdiff, bzegrep (link to bzgrep), bzfgrep (link to bzgrep), bzgrep, bzip2, bzip2recover, bzless (link to bzmore), and bzmore
Installerade bibliotek: libbz2.{a,so}
Installerade mappar: /usr/share/doc/bzip2-1.0.8

Korta beskrivningar

bunzip2	Dekomprimerar bzipped filer
bzcat	Dekomprimerar till standard output
bzcmp	Kör **cmp** på bzipped filer
bzdiff	Kör **diff** på bzipped filer
bzegrep	Kör **egrep** på bzipped filer
bzfgrep	Kör **fgrep** på bzipped filer
bzgrep	Kör **grep** på bzipped filer
bzip2	Komprimerar filer genom att använda algoritmen Burrows-Wheeler (blocksorterande text-komprimerande) med Huffman kodning: kompressionstakten är bättre än den vilken uppnås av mer traditionella komprimerare vilka använder “Lempel-Ziv”-algoritmer, som gzip.
bzip2recover	Försöker återskapa data från skadade bzippade filer.
bzless	Kör **less** på bzipped filer
bzmore	Kör **more** på bzipped filer
`libbz2`	Biblioteke som implementerar lossless, blocksorterande datakomprimering, genom att använda Burrows-Wheeler algoritmen.

6.13. Xz-5.2.4

Paketet Xz innehåller program för att komprimera och dekomprimera filer. Det tillhandahåller funktionalitet för format som Izma samt det nyare Xz. Att komprimera textfiler med hjälp av xz leder till ett procentuellt sett bättre resultat för komprimering, än med de traditionella gzip och bzip2 kommandona.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 16 MB

6.13.1. Installation av Xz

Förbered kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.2.4
```

Kompilera:

```
make
```

Testa:

```
make check
```

Installera paketet, och säkerställ att de nödvändiga filerna är installerade i de korrekta platserna:

```
make install
mv -v   /usr/bin/{lzma,unlzma,lzcat,xz,unxz,xzcat} /bin
mv -v /usr/lib/liblzma.so.* /lib
ln -svf ../../lib/$(readlink /usr/lib/liblzma.so) /usr/lib/liblzma.so
```

6.13.2. Innehåll Xz

Installerade program: lzcat (link to xz), lzcmp (link to xzdiff), lzdiff (link to xzdiff), lzegrep (link to xzgrep), lzfgrep (link to xzgrep), lzgrep (link to xzgrep), lzless (link to xzless), lzma (link to xz), lzmadec, lzmainfo, lzmore (link to xzmore), unlzma (link to xz), unxz (link to xz), xz, xzcat (link to xz), xzcmp (link to xzdiff), xzdec, xzdiff, xzegrep (link to xzgrep), xzfgrep (link to xzgrep), xzgrep, xzless, and xzmore
Installerade bibliotek: liblzma.so
Installerade mappar: /usr/include/lzma and /usr/share/doc/xz-5.2.4

Korta Beskrivningar

lzcat Dekomprimerar till standard output

lzcmp Kör **cmp** på LZMA komprimerade filer

lzdiff Kör **diff** på LZMA komprimerade filer

lzegrep Kör **egrep** på LZMA komprimerade filer

lzfgrep Kör **fgrep** på LZMA komprimerade filer

lzgrep Kör **grep** på LZMA komprimerade filer

lzless	Kör **less** på LZMA komprimerade filer
lzma	Komprimerar eller dekomprimerar filer med hjälp av LZMA formatet
lzmadec	En liten och snabb dekoder för LZMA komprimerade filer
lzmainfo	Visar information lagrad för LZMA komprimerade fil-headers
lzmore	Kör **more** på LZMA komprimerade filer
unlzma	Dekomprimerar filer med hjälp av LZMA formatet
unxz	Dekomprimerar filer med hjälp av XZ formatet
xz	Dekomprimerar eller komprimerar filer med hjälp av XZ formatet
xzcat	Dekomprimerar till standard output
xzcmp	Kör **cmp** på XZ komprimerade filer
xzdec	En liten och snabb dekoder för XZ komprimerade filer
xzdiff	Kör **diff** på XZ komprimerade filer
xzegrep	Kör **egrep** på XZ komprimerade filer
xzfgrep	Kör **fgrep** på XZ komprimerade filer
xzgrep	Kör **grep** på XZ komprimerade filer
xzless	Kör **less** på XZ komprimerade filer
xzmore	Kör **more** på XZ komprimerade filer
`liblzma`	Ett bibliotek som implementerar lossless, block-sorterande data komprimering, genom att använda Lempel-Ziv-Markov chain algoritmen.

6.14. File-5.38

Paketet File innehåller ett verktyg för att avgöra typen av en given fil eller flera filer.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 20 MB

6.14.1. Installation av File

Förbered för kompilering:

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

Kompiler:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

6.14.2. Innehåll File

Installerade program: file
Installerade bibliotek: libmagic.so

Short Descriptions

file Försöker klassificera filer; den gör detta igenom att köra ett antal test —file system test, magic number test, language test.

`libmagic` Innehåller rutiner för magic number igenkänning, vilka används av **file.**

6.15. Readline-8.0

Paketet Readline är ett antal bibliotek vilka tillhandahåller kommandolinje redigering och historik funktionalitet.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 15 MB

6.15.1. Installation av Readline

Ominstallation av Readline kommer leda till att gamla bibliotek flyttas till <libraryname>.old. Medan normalt sett detta inte betraktas som ett problem, kan det i vissa fall trigga en länkningsbugg i ldconfig. Nedan kommandon undviker detta:

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

Förbered för kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/readline-8.0
```

Kompilera:

```
make SHLIB_LIBS="-L/tools/lib -lncursesw"
```

Vad flaggorna betyder:

`SHLIB_LIBS="-L/tools/lib -lncursesw"`
Detta tvingar Readline att länka gentemot `libncursesw` biblioteket.

Detta paketet kommer inte med en testsvit.

Installera:

```
make SHLIB_LIBS="-L/tools/lib -lncursesw" install
```

Flytta nu de dynamiska biblioteket till en mer passande plats. Sätt även upp rättigheter och symboliska länkar:

```
mv -v /usr/lib/lib{readline,history}.so.* /lib
chmod -v u+w /lib/lib{readline,history}.so.*
ln -sfv ../../lib/$(readlink /usr/lib/libreadline.so) /usr/lib/libreadline.so
ln -sfv ../../lib/$(readlink /usr/lib/libhistory.so ) /usr/lib/libhistory.so
```

Om du önskar, installera då även dokumentationen:

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.0
```

6.15.2. Innehåll Readline

Installerade bibliotek: libhistory.so och libreadline.so
Installerade mappar: /usr/include/readline och /usr/share/doc/readline-8.0

Korta beskrivningar

`libhistory` Tillhandahåller ett användar-gränssnitt för att återkalla kommandotolkens linjer från dess lagring.

`libreadline` Tillhandahåller ett antal kommandon för att manipulera text vilken skrivs in vid ett programs interaktiva session.

6.16. M4-1.4.18

Paketet M4 innehåller en makro-processor.

Förväntad byggtid: 0.4 SBU
Nödvändigt disk-utrymme: 33 MB

6.16.1. Installation av M4

Gör först några förändringar som krävs av glibc-2.28:

```
sed -i 's/IO_ftrylockfile/IO_EOF_SEEN/' lib/*.c
echo "#define _IO_IN_BACKUP 0x100" >> lib/stdio-impl.h
```

Förbered för kompilering:

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

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

6.16.2. Innehåll M4

Installerade program: m4

Korta beskrivningar

m4 Kopierar de givna filerna medan expanderande de makron vilka de innehåller. Dessa makron är antingen inbyggda Eller användar-definierade och kan ta vilket antal argument som helst. Förutom att genomföra makro-expandering Har M4 inbyggda funktioner för att inkludera namngivna filer, köra unix-kommandon, genomföra integer aritmetik, Manipulera text, rekursion, etc. Programmet M4 kan användas antingen som en front-end för en kompilator eller Som en självständig makro-processor.

6.17. Bc-2.5.3

Paketet Bc innehåller ett arbiträrt precisions-numeriskt process språk.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 2.9 MB

6.17.1. Installation av Bc

Förbered för kompilering:

```
PREFIX=/usr CC=gcc CFLAGS="-std=c99" ./configure.sh -G -O3
```

Vad flaggorna betyder:

`CC=gcc CFLAGS="-std=c99"`
Dessa parametrar specifierar vilken kompilator samt vilken C-standard som skall användas.

`-O3`
Specifierar optimiseringen som skall användas.

`-G`
Utesluter delar av test-sviten. Dessa delar fungerar inte utan att GNU Bc finns tillgänglig.

Kompilera:

```
make
```

Kör test:

```
make test
```

Installera:

```
make install
```

6.17.2. Innehåll Bc

Installerade program: bc och dc

Korta beskrivningar

bc En kommando-linje kalkylator.

dc En reverse-polish kommando-linje kalkylator.

6.18. Binutils-2.34

Paketet Binutils innehåller en länkare, en assembler, samt andra verktyg för att hantera objektfiler.

Förväntad byggtid: 6.7 SBU
Nödvändigt disk-utrymme: 5.1 GB

6.18.1. Installation av Binutils

Verifiera att PTYs fungerar korrekt inom chroot-miljön, genom att köra nedan test:

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

Output av detta borde vara följande:

```
spawn ls
```

Om istället output motsvarar det vilket visas nedan, då är inte miljön anpassad för korrekt PTY användande. Detta är något vilket måste lösas innan testsviterna för Binutils och GCC körs.

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

Ta nu bort ett av testen, vilket hindar testsviten från att slutföras:

```
sed -i '/@\tincremental_copy/d' gold/testsuite/Makefile.in
```

Binutils dokumentation rekommenderar att Binutils byggs i en dedikerad byggmapp.

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

Förbered för kompilering:

```
../configure --prefix=/usr       \
             --enable-gold       \
             --enable-ld=default \
             --enable-plugins    \
             --enable-shared     \
             --disable-werror    \
             --enable-64-bit-bfd \
             --with-system-zlib
```

Vad flaggorna betyder:

`--enable-gold`
Bygg gold länkaren och installera den som ld.gold (bredvid standardlänkaren).

`--enable-ld=default`
Bygg originalet av bfd länkaren och installera den både som ld (standardlänkaren) och som ld.bfd.

`--enable-plugins`
Aktiverar plugin-support för länkaren.

`--enable-64-bit-bfd`
Aktiverar 64-bit support på värdar med mer begränsade ordstorlekar. Möjligen inte nödvändigt på 64-bit system, men skadar inte att aktivera.

`--with-system-zlib`
Använd det installerade zlib paketet, istället för att bygga den inkluderade versionen.

Kompilera:

```
make tooldir=/usr
```

Vad flaggorna betyder:

`tooldir=/usr`
Vanligtvis så är tooldir (mappen där exekverbara filer slutligen kommer att finnas) satt till $ (exec_prefix) / $ (targ et_alias). Exempelvis x86_64 maskiner skulle expandera detta till /usr/x86_64-unknown-linux-gnu. Eftersom detta är ett självbyggt system är denna target-specifika mapp i /usr inte av behov.
$(exec_prefix) / $ (target_alias) skulle användas ifall systemet användes för att cross-kompilera (exempelvis genom att kompilera ett paket på en Intel maskin, vilket generar kod som kan exekveras på PowerPC maskiner).

Viktigt

Test-sviten för Binutils betraktas i denna sektion som kritisk. Hoppa absolut inte över den.

Kör test:

```
make -k check
```

Tests ver_test_pr16504.sh är känt för att misslyckas, och du kan ignorera detta.

Installera:

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

6.18.2. Innehåll Binutils

Installerade program:	addr2line, ar, as, c++filt, dwp, elfedit, gprof, ld, ld.bfd, ld.gold, nm, objcopy, objdump, ranlib, readelf, size, strings, och strip
Installerade bibliotek:	libbfd.{a,so} och libopcodes.{a,so}
Installerade mappar:	/usr/lib/ldscripts

Korta beskrivningar

addr2line	Översätter programadresser till filnamn och och radnummer; given en adress samt namnet av en exekverbar fil, använder programmet felsökningsinformation i filen för att avgöra vilken source-fil samt radnummer som associeras med adressen.
ar	Skapar, modifierar, och extraherar från arkiv.
as	En assembler som skapar objektfiler från gcc output.
c++filt	Används av länkaren för att de-mangle C++ och Java symboler och för att undvika att funktioner vilka är overloaded kraschar med varandra.
dwp	DWARFs paketering-verktyg

elfedit	Uppdaterar ELF-filers ELF-header
gprof	Visar call-grafers profildata
ld	En länkare vilken kombinerar ett antal objekt och arkivfiler till en enstaka fil, omlokaliserande deras data och knytande samman symboliska referenser.
ld.gold	En minimiserad version av ld, vilken endast stödjer objektfils formatet ELF.
ld.bfd	Hård länk till **ld**
nm	Listar symbolerna i en given objektfil
objcopy	Översätter en typ av objektfil in till en annan
objdump	Visar information om den givna objektfilen, med alternativ vilka kontrollerar den specifierade info vilken skall visas. Informationen vilken visas är av nytta för programmerare vilka arbetar på de kompilerande verktygen.
ranlib	Generar ett index över innehållet av ett arkiv och lagrar det i arkivet. Indexet listar alla av symbolerna definierade av arkivmedlemmar vilka är objektfiler vilka kan omplaceras.
readelf	Visar information om ELF typ binärer
size	Listar sektionsstorlekarna samt den totala storleken av objektfiler
strings	Skriver ut, för varje given fil, sekvensen av skrivbara karaktärer vilka är åtminstone av den längd vilken specifierats (fyra som standard). För objektfiler skriver den som standard endast strängarna från de initialiserande och laddande sektionerna, medan för andra typer av filer den skannar hela filen.
strip	Raderar symboler från objektfiler
`libbfd`	Biblioteket: Binary File Descriptor
`libctf`	Felsöknings supprt för biblioteket: Compat ANSI-C Type Format
`libctf-nobfd`	En libctf variant vilken inte använder libbfd funktionalitet
`libopcodes`	Ett bibliotek för att hantera opcodes—de “läsbar text” versionerna av instruktioner till processorn. används för att bygga verktyg som **objdump**.

6.19. GMP-6.2.0

Paketet GMP innehåller matematikbibliotek. Dessa innehåller användbara funktioner för arbiträr precisionsaritmetik.

Förväntad byggtid: 1.1 SBU
Nödvändigt disk-utrymme: 51 MB

6.19.1. Installation of GMP

Kommentar

Om du bygger för 32-bit x86, men har en processor som är kapabel att köra 64-bit code *och* du har specifierat CFLAGS i miljön, då kommer configure skriptet försöka att konfigurera för 64-bit och fallera. Undvik detta genom att köra configure skriptet nedan med följande:

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

Kommentar

Standard inställningarna för GMP producerar bibliotek optimiserade för värdprocessorn. Om bibliotek vilka är lämpade för processorer mindre kapabla än värdens processor önskas, kan generiska bibliotek skapas med hjälp av nedan kommandon:

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

Förbered för kompilering:

```
./configure --prefix=/usr    \
            --enable-cxx     \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.2.0
```

Vad flaggorna betyder:

`--enable-cxx`
Aktiverar support för C++

`--docdir=/usr/share/doc/gmp-6.2.0`
Specifierar korrekt plats för dokumentationen

Kompilera paketet, samt generera HTML-dokumentation:

```
make
make html
```

Viktigt

Testsviten för GMP är kritisk. Hoppa inte över körandet av den.

Kör test:

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

Varning

Koden i GMP är kraftigt optimiserad för processorn vilken byggt GMP. Ibland så misslyckas koden vilken identifierar processorn att korrekt analysera systemets förmågor, vilket leder till fallerande test eller att program vilka använder GMP returnerar “Illegal instruction”. Om detta sker, bör GMP rekonfigureras med alternativet –build=x86_64-unknown-linux-gnu och byggas om.

Säkerställ att alla 190 test lyckades. Kolla resultaten genom att köra:

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

Installera paketet och dess dokumentation:

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

6.19.2. Innehåll GMP

Installerade bibliotek: libgmp.so och libgmpxx.so
Installerade mappar: /usr/share/doc/gmp-6.2.0

Korta beskrivningar

`libgmp` Innehåller precisions matematiska funktioner

`libgmpxx` Innehåller C++ precisions matematiska funktioner

6.20. MPFR-4.0.2

Paketet MPFR innehåller funktioner för multipel precisions matematik.

Förväntad byggtid: 0.8 SBU
Nödvändigt disk-utrymme: 37 MB

6.20.1. Installation av MPFR

Förbered för kompilering:

```
./configure --prefix=/usr        \
            --disable-static     \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.0.2
```

Kompilera paketet och generera dokumentationen:

```
make
make html
```

Viktigt

Testsviten för MPFR är kritisk. Hoppa inte över körandet av den.

Kör test och kolla så att alla test lyckades:

```
make check
```

Installera paketet och dess dokumentation:

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

6.20.2. Innehåll MPFR

Installerade bibliotek: libmpfr.so
Installerade mappar: /usr/share/doc/mpfr-4.0.2

Korta beskrivningar

`libmpfr` Innehåller multipel-precisions matematiska funktioner

6.21. MPC-1.1.0

Paketet MPC innehåller ett bibliotek för aritmetik av komplexa nummer, med arbiträr hög precision och korrekt avrundning av resultatet.

Förväntad byggtid: 0.3 SBU
Nödvändigt disk-utrymme: 22 MB

6.21.1. Installation av MPC

Förbered kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.1.0
```

Kompilera paketet och generera dokumentationen:

```
make
make html
```

Kör test:

```
make check
```

Installera paketet och dess dokumentation:

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

6.21.2. Innehåll MPC

Installerade bibliotek: libmpc.so
Installerade mappar: /usr/share/doc/mpc-1.1.0

Korta beskrivningar

`libmpc` Innehåller komplexa matematiska funktioner

6.22. Attr-2.4.48

Paketet attr innehåller verktyg för att administrera utvidgade attribut på filsystemobjekt.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 4.2 MB

6.22.1. Installation av Attr

Förbered för kompilering:

```
./configure --prefix=/usr     \
            --bindir=/bin     \
            --disable-static  \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.4.48
```

Kompilera paketet:

```
make
```

Testen måste köras på ett filsystem vilket stödjer utvidgade attribut såsom ext2, ext3, eller ext4 filsystem.
Kör testen:

```
make check
```

Installera paketet:

```
make install
```

Det delade biblioteket behöver flyttas till /lib, och som ett resultat av detta behöver .so filen i /usr/lib återskapas:

```
mv -v /usr/lib/libattr.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libattr.so) /usr/lib/libattr.so
```

6.22.2. Innehåll Attr

Installerade program: attr, getfattr, och setfattr
Installerade bibliotek: libattr.so
Installerade mappar: /usr/include/attr och /usr/share/doc/attr-2.4.48

Korta beskrivningar

attr Utvidgar attribut på filsystemsobjekt

getfattr Gets de utvidgade attributen på filsystemsobjekt

setfattr Sets de utvidgade attributen på filsystemsobjekt

`libattr` Innehåller biblioteksfunktioner för att manipulera utvidgade attribut

6.23. Acl-2.2.53

Paketet Acl innehåller verktyg för att administrera Access Control Lists, vilka används för att definiera mer detaljerade tillgångsrättigheter för filer och mappar.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 6.4 MB

6.23.1. Installation av Acl

Förbered för kompilering:

```
./configure --prefix=/usr         \
            --bindir=/bin         \
            --disable-static      \
            --libexecdir=/usr/lib \
            --docdir=/usr/share/doc/acl-2.2.53
```

Kompilera paketet:

```
make
```

ACLs test behöver köras på ett filsystem som stödjer access controls efter det att Coreutils har byggts med hjälp av ACL biblioteken. Om önskat, återvänd till detta paket och kör **make check** efter att coreutils har installerats i senare kapitel.

Installera paketet:

```
make install
```

Det delade biblioteket behöver flyttas till /lib, och som ett resultat av detta behöver .so filen i /usr/lib återskapas:

```
mv -v /usr/lib/libacl.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libacl.so) /usr/lib/libacl.so
```

6.23.2. Contents of Acl

Installerade program: chacl, getfacl, och setfacl
Installerade bibliotek: libacl.so
Installerade mappar: /usr/include/acl och /usr/share/doc/acl-2.2.53

Korta beskrivningar

chacl Modiferar access control list för en fil eller mapp

getfacl Gets file access control lists

setfacl Sets file access control lists

`libacl` Innehåller biblioteksfunktionerna för att manipulera Access Control Lists

6.24. Shadow-4.8.1

Paketet Shadow innehåller program för att hantera lösenord på ett säkert sätt och vis.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 46 MB

6.24.1. Installation av Shadow

Kommentar

Ifall du vill kräva användningen av säkra lösenord, referera då till *http://www.linuxfromscratch.org/blfs/view/* för att installera CrackLib innan det att du bygger Shadow. Lägg sedan till --with-libcrack till configure kommandona nedan.

Avaktivera installationen av **groups** programmet och dess man pages, eftersom Coreutils tillhandahåller en bättre version. Undvik även installationen av man pages vilka redan installerats i sektion 6.8, “Man-pages-5.05”:

```
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 / /'   {} \;
```

Istället för att använda standardmetoden *crypt*, använd den säkrare *SHA-512* för lösenordsenkryptering, vilken även tillåter lösenord längre än 8 symboler. Det är även nödvändigt att ändra den utgångna /var /spool /mail positionen för Användar-mailboxar som Shadow använder som standard, till /var /mail:

```
sed -i -e 's@#ENCRYPT_METHOD DES@ENCRYPT_METHOD SHA512@' \
       -e 's@/var/spool/mail@/var/mail@' etc/login.defs
```

Kommentar

Om du väljer att bygga Shadow med Cracklib support, kör följande:

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

Gör en mindre förändring så att det första gruppnummer som genereras av useradd blir 1000:

```
sed -i 's/1000/999/' etc/useradd
```

Förbered för kompilering:

```
./configure --sysconfdir=/etc --with-group-name-max-length=32
```

Vad de olika alternativen betyder:

`--with-group-name-max-length=32`
Användarnamn kan som längst vara 32 symboler långt. Gör så att detta även gäller gruppnamnen.

Kompilera paketet:

```
make
```

Detta paket inkluderar inte en testsvit.

Installera paketet:

```
make install
```

6.24.2. Konfigurering av Shadow

Detta paket innehåller verktyg för att skapa, modifiera, och radera användare samt grupper; definiera samt ändra deras lösenord; samt utföra diverse andra administrativa uppgifter. För en fullständig beskrivning av *shadow passwording*, läs doc/HOWTO filen i det ouppackade source trädet. Vid användning av Shadow support, kom ihåg att program vilka behöver verifiera lösenord(display managers, FTP program, pop3 daemoner, etc) måster vara Shadow kompatibla.

För att aktivera shadowed lösenord, kör:

```
pwconv
```

För att aktivera shadowed grupper, kör:

```
grpconv
```

Shadow's stock configuration för verktyget **useradd** innebär ett antal omständigheter vilka behöver förklaras, Först och främst är standard för **useradd** att skapa en användare samt en grupp med samma namn som användaren. Som standard kommer userid(UID) och groupid(GID) att börja med 1000. Detta innebär, att ifall du inte bifogar parametrar till **useradd** kommer varje användare att tillhöra en unik grupp. Ifall detta inte är önskvärt, behöver du skicka med flaggan -g till **useradd**. Standard parametrarna lagras i filen /etc /default /useradd. Du kan behöver modifiera denna fil för att få den att passa dina specifika behov.

`/etc/default/useradd` Parameter-förklaringar

`GROUP=1000`

Denna parameter definierar början av gruppnumren i filen /etc/group. Du kan modifiera den till vad du än önskar. Notera att **useradd** aldrig kommer återanvända ett UID eller GID. Om numret som identifiera i parametern redan används, kommer det närmaste tillgängliga nummer efter detta att istället användas. Notera även, att ifall du inte har en grupp 1000 på ditt system, kommer du få ett meddelande “useradd: unknown GID 1000” första gången du kör **useradd** utan -g. Du kan ignorerar detta meddelande, och GID 1000 kommer att användas.

`CREATE_MAIL_SPOOL=yes`

Denna parameter leder till att **useradd** skapar en mailbox fil för den nyligen skapade användare. **useradd** kommer sätta gruppägarskapet av denna till gruppen “mail”, med 0660 rättigheter. Om du inte vill att dessa filer skall skapas av **useradd**, kör följande kommando:

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

6.24.3. Att skapa lösenord för root

Välj ett lösenord för användaren *root* och implementera det med:

```
passwd root
```

6.24.4. Contents of Shadow

Installed programs: chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgidmap, newgrp, newuidmap, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (link to newgrp), su, useradd, userdel, usermod, vigr (link to vipw), and vipw

Installed directory: /etc/default

Short Descriptions

chage Används för att ändra det maximala antalet dagar emellan obligatoriska ändringar av lösenord

chfn Används för att ändra en användares fullständiga namn, samt annan information

chgpasswd Används för att uppdatera grupp-lösenord i batch mode

chpasswd Används för att uppdatera användar-lösenord i batch mode

chsh Används för att ändra en användares standard-skal för inloggning

expiry Kollar och ser till att den nuvarande lösenords-policyn efterlevs

faillog Används för att undersöka loggen för inloggnings-misslyckanden; för att sätta ett max antal av försök att logga in; eller för att återställa antalet gånger som försök till inloggning har misslyckats.

gpasswd Används för att radera och lägga till medlemmar till en grupp.

groupadd Skapar en grupp med givet namn

groupdel Raderar en grupp med givet namn

groupmems Tillåter en användare att administrera sitt eget grupp-medlemskap, utan kravet av super user privilegier.

groupmod Används för att modifiera den givna gruppens namn eller GID.

grpck Verifierar integriteten av de två filerna `/etc/group` och `/etc/gshadow`

grpconv Skapar eller uppdaterar - med hjälp av en normal gruppfil - shadow gruppfilen.

grpunconv Uppdaterar`/etc/group` från `/etc/gshadow` och raderar därefter den senare.

lastlog Rapporterar de senaste loginsen för lagrade användare, eller för en given sådan

login Används av systemet så att användare ska kunna logga in

logoutd Är en daemon som upprätthåller restriktioner på tid för inloggningar och portar

newgidmap Används för att sätta GID mappingen för ett användar-namespace

newgrp Används för att ändra det nuvarnde GID under en inloggad session

newuidmap Används för att sätta UID mappingen för ett användar-namespace

newusers Används för att skapa eller uppdatera en serie av användar-konton

nologin Visar ett meddelande att ett konto inte är tillgängligt. Designat för att vara standard-skalet för konton Vilka har inaktiverats.

passwd Används för att ändra lösenordet för en användare eller ett gruppkonto.

pwck Verifierar integriteten av lösenordsfilerna `/etc/passwd` och `/etc/shadow`

pwconv Skapar eller uppdaterar shadow lösenordsfilen med hjälp av den normala lösenordsfilen.

pwunconv Uppdaterar`/etc/passwd` via`/etc/shadow` och raderar därefter den senare.

sg	Utför ett givet kommando, medan det att användarens GID är satt till en given grupps GID
su	Kör ett skal med ersättnings IDn för användare och grupp.
useradd	Skapar en ny användare med givet namn, eller uppdaterar standard ny-användare informationen
userdel	Raderar ett givet användarkonto
usermod	Används för att modifiera den givna användarens namn för inloggning, UID, skal, startgrupp, /home, etc.
vigr	Redigerar `/etc/group`eller`/etc/gshadow` filerna
vipw	Redigerar `/etc/passwd` eller`/etc/shadow` filerna

6.25. GCC-9.2.0

Paketet GCC innehåller GNUs kompilatorer, vilket inkluderar C och C++ kompilatorerna.

Förväntad byggtid: 88 SBU (med test)
Nödvändigt disk-utrymme: 4.2 GB

6.25.1. Installation av GCC

Om du bygger på x86_64, ändra standard mappnamn för 64-bit bibliotek till “lib”:

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

Precis som då gcc byggdes en andra gång, fixa även nu problemet vilket introducerades av Glibc-2.31:

```
sed -e '1161 s|^|//|' \
    -i libsanitizer/sanitizer_common/sanitizer_platform_limits_posix.cc
```

Dokumentationen rekommenderar att gcc byggs i en dedikerad byggmapp:

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

Förbered för kompilering:

```
SED=sed                               \
../configure --prefix=/usr            \
             --enable-languages=c,c++ \
             --disable-multilib       \
             --disable-bootstrap      \
             --with-system-zlib
```

Notera att för andra språk, finns det vissa förkrav vilka ännu inte är tillgängliga. Läs BLFS för instruktioner vad gäller hur man bygger alla språk stödda av GCC.

Vad alternativen betyder:

`SED=sed`
Genom att definiera denna miljövariabel förhindrar man en hårdkodad sökväg till /tools/bin/sed.

`--with-system-zlib`
Definierar för GCC att länka till det systeminstallerade Zlib biblioteket, istället för dess egna interna kopia.

Kompilera paketet:

```
make
```

Viktigt

Testsviten i denna sektion betraktas som kritisk att köra. Hoppa inte över den.

Ett av testen i GCCs testsvit har för vana att använda upp hela stacken. Utöka därför stackens storlek innan testen körs:

```
ulimit -s 32768
```

Tesa resultaten som en opriviligerad användare, men stanna inte upp vid errors:

```
chown -Rv nobody .
su nobody -s /bin/bash -c "PATH=$PATH make -k check"
```

För att få en sammanfattning av testresultaten, kör:

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

För endast en sammanfattning, pipe output genom **`grep -A7 Summ`**

Resultat kan jämföras med de vilka finns på *http://www.linuxfromscratch.org/lfs/build-logs/9.1/* och *https://gcc.gnu.org/ml/gcc-testresults/*.

Sex test relaterade till get_time misslyckas ofta. Dessa är tydligen relaterade till locale en_HK.

Två test vid namn lookup.cc och reverse.cc i experimental/net misslyckas ofta i LFS chroot miljö eftersom de kräver /etc /hosts och iana-etc.

Två test vid namn pr57193.c och pr90178.c misslyckas ofta.

Ett mindre antal oväntade misslyckanden kan inte alltid undvikas. GCC-utvecklarna är vanligtvis medvetna om dessa problem, men har inte löst dem ännu. Ifall inte testresultaten avviker omfattande från de i länkarna ovan, bara fortsätt.

Installera paketet och ta bort en onödig mapp:

```
make install
rm -rf /usr/lib/gcc/$(gcc -dumpmachine)/9.2.0/include-fixed/bits/
```

GCCs byggmapp ägs av “nobody” nu, och ägarskapet av den installerade headermappen(och dess innehåll) kommer vara inkorrekt. Ändra ägarskap till användare och grupp root.

```
chown -v -R root:root \
    /usr/lib/gcc/*linux-gnu/9.2.0/include{,-fixed}
```

Skapa en symlänk vilken krävs av FHS till följd av "historiska" orsaker.

```
ln -sv ../usr/bin/cpp /lib
```

Många paket använder namnet **cc** för att kalla C kompilatorn. Tillfredställ dessa paket genom att skap en symlänk:

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

Skapa en kompatibilitets symlänk för att möjliggöra att bygga program med hjälp av Link Time Optimization(LTO):

```
install -v -dm755 /usr/lib/bfd-plugins
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/9.2.0/liblto_plugin.so \
        /usr/lib/bfd-plugins/
```

Då vi nu har vår slutgiltiga toolchain, är det viktigt att återigen säkerställa att kompilering och länking fungerar som det är tänkt. Kör därför samma koll som tidigare:

```
echo 'int main(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

Inga errors borde rapporteras, och output av det sista kommandot kommer vara (undantaget för plattformsspecifika skillnader i den dynamiska länkarens namn):

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

Säkerställ nu att allt är inställt så att de korrekta startfilerna används:

```
grep -o '/usr/lib.*/crt[1in].*succeeded' dummy.log
```

Output för det sista kommandot borde vara:

```
/usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/../../../../lib/crt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/../../../../lib/crtn.o succeeded
```

Beroende på din maskins arkitektur, kan texten ovan variera litet. Skillnaden är vanligtvis namnet på mappen efter /usr /lib /gcc. Det viktiga att kolla efter här, är att gcc har hittat alla tre crt*.o filerna under mappen /usr /lib.

Verifiera att kompilatorn söker efter de korrekta headerfilerna:

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

Detta kommando borde returnera följande output:

```
#include <...> search starts here:
 /usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/include-fixed
 /usr/include
```

Åter igen, notera att mappen vilken namngivits efter din target triplet, kan vara en annan än den vilken nämns ovan. Detta sker tillföljd av att du har en annan systemarkitektur.

Verifiera därefter att den nya länkaren använder de korrekta sökvägarna:

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

Referencer till sökvägar vilka har komponenter med '-linux-gnu' borde ignoreras, men bortsett från detta borde output för det sista kommandot vara:

```
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");
```

Ett 32-bit system kan innebära ett antal andra mappat. Här nedan är till exempel output från 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");
```

Säkerställ därefter att den korrekta versionen av Libc används:

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

Output för detta kommando borde vara:

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

Säkerställ sedan att gcc använder den korrekta dynamiska länkaren:

```
grep found dummy.log
```

Output för detta kommando borde vara (med undantagn för plattforms-specifika skillnader i länkarens namn):

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

Om output inte visas som kan ses ovan eller inte visas alls, är något allvarligt fel. Felsök baklänges steg för steg, tills du hittar vilket felet är och kan fixa det. Den mest sannolika orsaken är att något gick fel vid modifikation av specifikationsfilen. Problem här måste lösas innan det att du fortsätter vidare i boken.

Då allt väl fungerar som det är tänkt, städa upp testfilerna:

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

Flytta slutligen en fil som befinner sig på fel plats:

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

6.25.2. Innehåll GCC

Installerade program:	c++, cc (link to gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, gcov, gcov-dump, och gcov-tool
Installerade bibliotek:	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}, och libubsan.{a,so}
Installerade mappar:	/usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc, och /usr/share/gcc-9.2.0

Korta beskrivningar

c++	C++ kompilatorn

cc	C kompilatorn
cpp	C preprocessor; används av kompilatorn för att expander #include, #define, och liknande definitioner i källkodsfilerna.
g++	C++ kompilatorn
gcc	C kompilatorn
gcc-ar	En wrapper omkring **ar** som tillför en plugin till kommandolinjen. Detta program används endast för att lägga till "link time optimization" och är inte relevant för standard byggalternativen.
gcc-nm	En wrapper omkring **nm** som tillför en plugin till kommandolinjen. Detta program används endast för att lägga till "link time optimization" och är inte relevant för standard byggalternativen.
gcc-ranlib	En wrapper omkring **ranlib** som tillför en plugin till kommandolinjen. Detta program används endas för att lägga till "link time optimization" och är inte relevant för standard byggalternativen.
gcov	Ett coverage testing tool; det används för att analysera program så att det går avgöra var det går att implementera optimiseringar så att de får största möjliga effekt.
gcov-dump	Offline gcda och gcno profile dump tool
gcov-tool	Offline gcda profile processing tool
`libasan`	Address Sanitizer runtime bibliotek
`libatomic`	GCC atomic inbyggt runtime bibliotek
`libcc1`	C preprocessing bibliotek
`libgcc`	Innehåller run-time stöd för **gcc**
`libgcov`	Detta bibliotek är länkat in i ett program då GCC instrueras att aktivera profilering
`libgomp`	GNU implementation av OpenMP API för multi-plattform delat-minne paralell programmering i C/C++ och Fortran
`liblsan`	Leak Sanitizer runtime bibliotek
`liblto_plugin`	GCC's Link Time Optimization (LTO) plugin möjliggör för GCC att utföra optimiseringar för kompileringsenheter.
`libquadmath`	GCC Quad Precision Math bibliotek API
`libssp`	Innehåller rutiner som stödjer GCC's stack-smashing protection funktionalitet
`libstdc++`	Standard C++ biblioteket
`libstdc++fs`	ISO/IEC TS 18822:2015 filsystems bibliotek
`libsupc++`	Innehåller stödjande rutiner för C++
`libtsan`	Thread Sanitizer runtime bibliotek
`libubsan`	Undefined Behavior Sanitizer runtime bibliotek

6.26. Pkg-config-0.29.2

Paketet pkg-config innehåller ett verktyg vilket vidarebefordrar include path och/eller sökvägar till byggverktyg, under configure och make file exekvering.

Förväntad byggtid: 0.3 SBU
Nödvändigt disk-utrymme: 30 MB

6.26.1. Installation av Pkg-config

Förbered för kompilering:

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

Vad alternativen betyder:

`--with-internal-glib`
Tillåter pkg-config att använda sin interna version av Glib eftersom en extern version inte är tillgänglig i LFS.

`--disable-host-tool`
Avaktiverar skapandet av en oönskad hårdlänk till pkg-config programmet.

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

6.26.2. Innehåll Pkg-config

Installerade program: pkg-config
Installerade mappar: /usr/share/doc/pkg-config-0.29.2

Korta beskrivningar

pkg-config Returnerar meta information för det specifierade biblioteket eller paketet

6.27. Ncurses-6.2

Paketet Ncurses innehåller bibliotekt för terminal-oberoende hantering av karaktärs skärmar.

Förväntad byggtid: 0.4 SBU
Nödvändigt disk-utrymme: 43 MB

6.27.1. Installation av Ncurses

Installera inte ett statiskt bibliotek vilket inte är hanterat av configure:

```
sed -i '/LIBTOOL_INSTALL/d' c++/Makefile.in
```

Förbered för kompilering:

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

The meaning of the new configure options:

`--enable-widec`
Leder till att wide-character bibliotek (t.ex `libncursesw.so.6.2`)byggs istället för de vanliga varianterna (`libncurses.so.6.2.` Dessa bibliotek är användbara i båda multibyte och traditionella 8bit locales, medan vanliga bibliotek endast fungerar korrekt i 8bit locales. Wide-character och vanliga bibliotek är källkods-kompatibla men inte binär-kompatibla.

`--enable-pc-files`
Generar och installerar .pc files för pkg-config.

`--without-normal`
Deaktiverar byggandet och installerandet av de flesta statiska bibliotek.

Kompilera:

```
make
```

Detta paket har en test-svit, men den kan endast köras efter det att paketet har installerats. Testen återfinns i mappen test/ . För mer information, läs README filen i denna mapp.

Installera:

```
make install
```

Flytta de delade biblioteken till mappen /lib, där det är förväntat att de ska existera.

```
mv -v /usr/lib/libncursesw.so.6* /lib
```

Eftersom biblioteken har flyttats pekar en av symlänkarna till en icke existent fil. Återskapa denna fil.

```
ln -sfv ../../lib/$(readlink /usr/lib/libncursesw.so) /usr/lib/libncursesw.so
```

Många applikationer förväntar sig fortfarande att länkaren är kapabel att finna non-wide-character Ncurses bibliotek. Lura sådana applikationer till att länka med wide-character bibliotek genom att använda symlänkar och länkar-scripts.

```
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
```

Slutligen, säkerställ att gamla applikationer vilka letar efter -lcurses vid byggtid fortfarande är byggbara.

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

Om önskat, installera Ncurses dokumentation:

```
mkdir -v       /usr/share/doc/ncurses-6.2
cp -v -R doc/* /usr/share/doc/ncurses-6.2
```

Kommentar

Instruktionerna ovan skapar inte non-wide-character Ncurses bibliotek eftersom inget paket installerat genom att kompilera från källkod skulle länka gentemot dem under runtime. Däremot, de enda kända endast-binära applikationer vilka länkar gentemot non-wide-character Ncurses bibliotek kräver version 5. Om du måste ha sådana bibliotek på grund av någon typ av endast-binär applikation eller för att följa LBS regleringar, bygg paketet igen med de följande kommandona:

```
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
```

6.27.2. Innehåll Ncurses

Installerade program: captoinfo (link to tic), clear, infocmp, infotocap (link to tic), ncursesw6-config, reset (link to tset), tabs, tic, toe, tput, och tset

Installerade bibliotek: libcursesw.so (symlänk och linker script till libncursesw.so), libformw.so, libmenuw.so, libncursesw.so, libncurses++w.a, libpanelw.so, och deras non-wide-character counterparts utan "w" i biblioteksnamnen.

Installerade mappar: /usr/share/tabset, /usr/share/terminfo, och /usr/share/doc/ncurses-6.2

Korta beskrivningar

captoinfo Omvandlar en termcap beskrivning till en terminfo beskrivning

clear	Rensar skärmen, om möjligt
infocmp	Jämför eller skriver ut terminfo beskrivningar
infotocap	Omvandlar en terminfo beskrivning till en termcap beskrivning
ncursesw6-config	Tillhandahåller konfigurationsinformation till ncurses
reset	Återinitialiserar en terminal till dess standard värden
tabs	Rensar samt sätter tab stops på en terminal
tic	Terminfo entry-description kompilatorn som översätter en terminfo fil från källkods format in till det binära format vilket behövs för ncurses biblioteks rutiner. En terminfo fil innehåller information vad gäller en enskild terminals förmågor.
toe	Listar alla tillgängliga terminaltyper, givande namn och beskrivning för varje av dessa.
tput	Gör värdena av terminal-beroende förmågor tillgängliga för skalet; kan också användas för att återsälla eller initialisera en terminal eller rapportera vad dess långa namn är.
tset	Kan användas för att initialisera terminaler
`libcursesw`	Länk till `libncursesw`
`libncursesw`	Innehåller funktioner för att visa text på många komplexa sätt på en terminal-skärm. Ett bra exempel vad gäller användandet av dessa funktioner är menyn vilka visas vid kernelns **make menuconfig**
`libformw`	Innehåller funktioner för att implementera former
`libmenuw`	Innehåller funktioner för att implementera menyer
`libpanelw`	Innehåller funktioner för att implementera paneler

6.28. Libcap-2.31

Paketet Libcap implementerar user-space gränsnittet med POSIX 1003.1e förmågorna tillgängliga i Linux kernels. Dessa förmågor innebär en partitionering av de kraftulla root privilegierna, in till ett set av distinkta privilegier.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 8.5 MB

6.28.1. Installation av Libcap

Förhindra två statiska bibliotek från att installeras:

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

Kompilera:

```
make lib=lib
```

Vad alternativen betyder:

`lib=lib`
Denna parameter sätter biblioteksmappen till `/lib` istället för `/lib64` på x86_64. Den har ingen effekt på x86.

Kör test:

```
make test
```

Installera:

```
make lib=lib install
chmod -v 755 /lib/libcap.so.2.31
```

6.28.2. Innehåll Libcap

Installerade program: capsh, getcap, getpcaps, och setcap
Installerade bibliotek: libcap.so och libpsx.a

Korta beskrivningar

capsh	Ett skal wrapper för att utforska och begränsa stöd av förmågor
getcap	Undersöker filförmågor
getpcaps	Visar förmågorna på de köade processerna
setcap	Definierar filförmågor
`libcap`	Innehåller biblioteksfunktionerna för att manipulera POSIX 1003.1e förmågor
`libpsx`	Innehåller funktioner för att stödja POSIX semantik för syscalls relaterade till pthread biblioteket

6.29. Sed-4.8

Paketet Sed innehåller en stream redigerare.

Förväntad byggtid: 0.4 SBU
Nödvändigt disk-utrymme: 34 MB

6.29.1. Installation av Sed

Fixa först ett problem i LFS miljön, och ta sedan bort ett ickefungerande test:

```
sed -i 's/usr/tools/'                 build-aux/help2man
sed -i 's/testsuite.panic-tests.sh//' Makefile.in
```

Förbered för kompilering:

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

Kompilera paketet och generera dokumentationen:

```
make
make html
```

Kör test:

```
make check
```

Installera paketet och dess dokumentation:

```
make install
install -d -m755           /usr/share/doc/sed-4.8
install -m644 doc/sed.html /usr/share/doc/sed-4.8
```

6.29.2. Innehåll Sed

Installerade program: sed
Installerade mappar: /usr/share/doc/sed-4.8

Korta beskrivningar

sed Filtrerar och omvandlar textfiler

6.30. Psmisc-23.2

Paketet Psmisc innehåller program för att visa information om aktiva processer.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 4.6 MB

6.30.1. Installation av Psmisc

Förbered för kompilering:

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

Kompilera:

```
make
```

Detta paket kommer inte med en test-svit.

Installera:

```
make install
```

Slutligen, flytta progammen **killall** och **fuser** till destinationen specifierade av FHS:

```
mv -v /usr/bin/fuser   /bin
mv -v /usr/bin/killall /bin
```

6.30.2. Innehåll Psmisc

Installerade program: fuser, killall, peekfd, prtstat, pslog, pstree, och pstree.x11 (länk till pstree)

Korta beskrivningar

fuser	Rapporterar Process IDs (PIDs) av processer som använder givna filer eller filsystem
killall	Dödar processer utefter namn, genom att skicka en signal till alla processer som kör givet kommando
peekfd	Peek på file descriptors för en aktiv process, med hjälp av dess PID
prtstat	Skriver ut information om en process
pslog	Rapporterar nuvarnade logs sökväg för en process
pstree	Visar aktiva processer i trädformat
pstree.x11	Samma som **pstree**, men att det väntar på en bekräftelse innan att det stängs ned

6.31. Iana-Etc-2.30

Paketet Iana-Etc tillhandahåller data för nätverkstjänster och protokoll

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 2.3 MB

6.31.1. Installation av Iana-Etc

Följande kommando omvandlar den råa data tillhandahållen av IANA till korrekta format för `/etc/protocols` och /etc /services datafiler:

```
make
```

Detta paket kommer inte med en testsvit.

Installera:

```
make install
```

6.31.2. Innehåll av Iana-Etc

Installrade filer: /etc/protocols och /etc/services

Korta beskrivningar

`/etc/protocols` Beskriver de divese DARPA internet protokoll som är tillgängliga via TCP/IP subsystemet

`/etc/services` Tillhandahåller en mapping mellan läsvänliga textnamn för internettjänster, samt dess Underliggande portnummer och protokolltyper.

6.32. Bison-3.5.2

Paketet Bison innehåller en parser generator.

Förväntad byggtid: 0.3 SBU
Nödvändigt disk-utrymme: 43 MB

6.32.1. Installation av Bison

Förbered för kompilering:

```
./configure --prefix=/usr --docdir=/usr/share/doc/bison-3.5.2
```

Kompilera:

```
make
```

Det existerar ett cirkulärt beroende emellan Bison och Flex vad gäller make check. Om så önskas, kan man efter att ha installerat Flex i nästan sektion, återbygga Bison och köra Bison checks med **make check**.

Installera:

```
make install
```

6.32.2. Innehåll Bison

Installerade program: bison och yacc
Installerade bibliotek: liby.a
Installerade mappar: /usr/share/bison

Korta beskrivningar

bison Genererar, från en serie regler, ett program vilket analyserar strukturen för textfiler. Bison är ett ersättnings-program för YACC(Yet Another Compiler Compiler).

yacc En wrapper för **bison**, för program vilka använder **yacc** istället för **bison**; den kallar **bison** med flaggan -y

`liby` Biblioteket för YACC vilket innehåller implementationer av YACC-kompatibla yyerror och main funktioner. Detta bibliotek är vanligtvis inte så användbart, men POSIX kräver det.

6.33. Flex-2.6.4

Paketet Flex innehåller ett verktyg för att generar program vilka kan urskilja mönster i text.

Förväntad byggtid: 0.4 SBU
Nödvändigt disk-utrymme: 36 MB

6.33.1. Installation av Flex

Fixa först ett problem som introducerades av glibc-2.26:

```
sed -i "/math.h/a #include <malloc.h>" src/flexdef.h
```

Byggproceduren utgår ifrån att programmet help2man är tillgängligt för att skapa en man page från det exekverbara alternativet --help. Detta alternativ finns inte tillgängligt, så vi använder en miljövariabel för att skippa detta. Förbered nu:

```
HELP2MAN=/tools/bin/true \
./configure --prefix=/usr --docdir=/usr/share/doc/flex-2.6.4
```

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

Ett mindre antal program är inte medvetna om Flex ännu, och försöker köra sin egen predecessor, **lex**. För att stödja dessa program, skapa en symbolisk länk vi namn “lex” vilken kör Flex i **lex** emulations tillstånd:

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

6.33.2. Innehåll Flex

Installerade program: flex, flex++ (länk till flex), och lex (länk till flex)
Installerade bibliotek: libfl.so
Installerade mappar: /usr/share/doc/flex-2.6.4

Korta beskrivningar

flex Ett verktyg för att generar program som kan urskilja mönster i text. Det möjliggör att specifiera regler för mönsteridentifiering, vilket utrotar behovet att skapa ett specialiserat program för detta ändamål.

flex++ En förlängning av flex. Används för att generera C++ kod och klasser. Symbolisk länk till Flex

lex En symbolisk länk som kör **flex** i **lex** emulations tillstånd

`libfl` Biblioteket för flex

6.34. Grep-3.4

Paketet Grep innehåller verktyg för att genomsöka filer

Förväntad byggtid: 0.7 SBU
Nödvändigt disk-utrymme: 39 MB

6.34.1. Installation av Grep

Förbered för kompilering:

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

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera :

```
make install
```

6.34.2. Innehåll Grep

Installerade program: egrep, fgrep, och grep

Korta beskrivningar

egrep Skriver rader vilka matchar ett förlängt regular expression

fgrep Skriver rader vilka matchar en lista av bestämda strängar

grep Skriver rader vilka matchar ett grundläggande regular expression

6.35. Bash-5.0

Paketet Bash innehåller Bourne-Again SHell.

Förväntad byggtid: 1.9 SBU
Nödvändigt disk-utrymme: 62 MB

6.35.1. Installation av Bash

Implementera några upstream fixar:

```
patch -Np1 -i ../bash-5.0-upstream_fixes-1.patch
```

Förbered för kompilering:

```
./configure --prefix=/usr                      \
            --docdir=/usr/share/doc/bash-5.0 \
            --without-bash-malloc              \
            --with-installed-readline
```

Vad alternativen betyder:

`--with-installed-readline`
Instruerar Bash att använda readline biblioteket vilket redan är installerat på systemet, istället för att använda sin egen readline version.

Kompilera:

```
make
```

Om du inte vill köra testen, gå direkt vidare till att installera paketet.

För att förbereda testen, säkerställ att användaren nobody kan skriva till /sources träd:

```
chown -Rv nobody .
```

Kör nu testen som användare `nobody`

```
su nobody -s /bin/bash -c "PATH=$PATH HOME=/home make tests"
```

Installera paketet och flytta main executable till `/bin` :

```
make install
mv -vf /usr/bin/bash /bin
```

Kör det nu installerade Bash (vilket ersätter det som för tillfället exekveras):

```
exec /bin/bash --login +h
```

Kommentar

Parametrarna som används gör processen **bash** till ett interaktivt inloggningsskal och fortsätter att inaktivera hashing så att nya program hittas allteftersom att de installeras.

6.35.2. Innehåll Bash

Installerade program: bash, bashbug, och sh (länk till bash)
Installed directory: /usr/include/bash, /usr/lib/bash, och /usr/share/doc/bash-5.0

Korta beskrivningar

bash En kommandotolk. Utför många olika typer av expansioner och ersättningar på en given kommandorad innan det att den exekverar raden. En kraftull kommandotolk.

bashbug Skalskript som hjälper användaren att sammanställa och maila in standardformaterade bugrapporter för bash

sh En symlänk till **bash**; när exekverad som **sh,** försöker **bash** att härma uppstarts beteendet av historiska versioner av **sh** i så stor utsträckning som möjligt, samtidigt som försöker förhålla sig till POSIX standarder.

6.36. Libtool-2.4.6

Paketet Libtool innehåller GNUs generiska bibliotek stödskript. Det wrappar komplexiteten av att använda delade bibliotek i ett sammanhängande portabelt gränssnitt.

Förväntad byggtid: 1.8 SBU
Nödvändigt disk-utrymme: 43 MB

6.36.1. Installation av Libtool

Förbered för kompilering:

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

Kompilera:

```
make
```

Kör test:

```
make check
```

Kommentar

Testtiden för libtool kan reduceras kraftigt på system med flertalet kärnor. För att göra detta, lägg till TESTSUITEFLAGS=-j<N> till raden ovan. Genom att använda t.ex -j4, kan man reducera testtiden med upp till 60%.

Fem test är kända för att misslyckas i LFS byggmiljö, på grund av ett cirkulärt beroende. Alla test lyckas ifall de körs efter att makecheck har installerats.

Installera:

```
make install
```

6.36.2. Innehåll Libtool

Installerade program: libtool och libtoolize
Installerade bibliotek: libltdl.so
Installerade mappar: /usr/include/libltdl och /usr/share/libtool

Korta beskrivningar

libtool Tillhandahåller generaliserade supporttjänster för biblioteksbyggande

libtoolize Tillhandahåller ett standardsätt för att lägga till **libtool** stöd till ett paket

`libltdl` Gömmer de diverse svårigheterna med dlopening bibliotek

6.37. GDBM-1.18.1

Paketet GDBM innehåller GNU Database Manager. Det är ett paket med databasfunktioner som använder extensible hashing och fungerar på ett liknande sätt som det standard UNIX dbm. Biblioteket tillhandahåller primitiva funktioner för att lagra nyckel/data par, sök och motta data med hjälp av dess nyckel, samt att radera en nyckel tillsammans med data.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 11 MB

6.37.1. Installation av GDBM

Förbered för kompilering:

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

Vad alternativen betyder:

`--enable-libgdbm-compat`
Möjliggör för libgdbm kompatibilitets bibliotek att byggas, då vissa paket utanför LFS kan kräva de äldre DMB rutiner vilket detta paket tillhandahåller.

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

6.37.2. Innehåll GDBM

Installerade program: gdbm_dump, gdbm_load, och gdbmtool
Installerade bibliotek: libgdbm.so och libgdbm_compat.so

Korta beskrivningar

gdbm_dump	Dumpar en GDBM databas till en fil
gdbm_load	Återskapar en GDBM databas från en dumpfil
gdbmtool	Testar och modifierar en GDBM databas
`libgdbm`	Innehåller funktioner för att manipulera en hashad databas
`libgdbm_compat`	Kompatibilitets bibliotek vilket innehåller äldre DBM funktioner

6.38. Gperf-3.1

Gperf genererar ett perfekt hash från ett set av nycklar.

Förväntad byggtid: mindre än 0.1 SBU
Nödvändigt disk-utrymme: 6.3 MB

6.38.1. Installation av Gperf

Förbered för kompilering:

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

Kompilera:

```
make
```

Testen kan misslyckas om man kör flera test samtidigt (-j högre än 1). För att köra testen:

```
make -j1 check
```

Installera:

```
make install
```

6.38.2. Innehåll Gperf

Installerade program: gperf
Installerade mappar: /usr/share/doc/gperf-3.1

Korta beskrivningar

gperf Generar ett perfekt hash från ett nyckelset

6.39. Expat-2.2.9

Paketet Expat innehåller ett stream orienterat C bibliotek för att parsea XML.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 11 MB

6.39.1. Installation av Expat

Fixa först ett problem med regressionstesten, vilket uppstår i LFS miljön:

```
sed -i 's|usr/bin/env |bin/|' run.sh.in
```

Förbered för kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.2.9
```

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

Om önskat, installera dokumentationen:

```
install -v -m644 doc/*.{html,png,css} /usr/share/doc/expat-2.2.9
```

6.39.2. Innehåll Expat

Installerade program: xmlwf
Installerade bibliotek: libexpat.so
Installerade mappar: /usr/share/doc/expat-2.2.9

Korta beskrivningar

xmlwf Ett icke-validerande verktyg för att kolla ifall XML-dokument är korrekt formade

`libexpat` Innehåller API funktions för att parsea XML

6.40. Inetutils-1.9.4

Paketet Inetutils innehåller program för grundläggande nätverksfunktionalitet.

Förväntad byggtid: 0.3 SBU
Nödvändigt disk-utrymme: 29 MB

6.40.1. Installation av Inetutils

Förbered för kompilering:

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

Vad alternativen betyder:

`--disable-logger`
Förhindrar inetutils från att installera programmet **logger**, vilket används av skript för att skicka meddelanden till System Log Daemon. Installera det inte, eftersom Util-linux installerar en nyare version.

`--disable-whois`
Deaktiverar byggandet av Inetutils **whois** klient, vilken är utdaterad. Instruktioner för en bättre **whois** klient återfinns i boken för BLFS.

`--disable-r*`
Dessa parametrar deaktiverar byggandet av utdaterade program vilka inte borde användas på grund av säkerhets-relaterade orsaker. Funktionerna tillhandahållna av dessa program kan tillhandahållas av openssh paketet i BLFS.

`--disable-servers`
Deaktiverar installationen av diverse nätverks-servers vilka inkluderas som del av Inetutils paketet. Dessa servrar betraktas som icke lämpliga i ett grundläggande LFS system. Vissa är av sin natur osäkra och är endast att betrakta som säkra ifall de körs på pålitliga nätverk. Bättre ersättningsservrar finns för de flesta av dessa servrar.

Kompilera:

```
make
```

Kör test:

```
make check
```

Kommentar

Ett test, libls.sh, kan misslyckas i den inledande chroot miljön men kommer lyckas ifall testet körs igen efter det att LFS byggts färdigt. Ping-localhost.sh, kommer misslyckas ifall värdsystem inte är ipv6 kompatibelt.

Installera:

```
make install
```

Flytta program, så de är tillgängliga även ifall`/usr` inte kan nås:

```
mv -v /usr/bin/{hostname,ping,ping6,traceroute} /bin
mv -v /usr/bin/ifconfig /sbin
```

6.40.2. Innehåll Inetutils

Installerade program: dnsdomainname, ftp, ifconfig, hostname, ping, ping6, talk, telnet, tftp, och traceroute

Korta beskrivningar

dnsdomainname	Visar systemets DNS domän namn
ftp	File transfer protocol program
hostname	Rapporterar eller definierar värdens namn
ifconfig	Hanterar nätverksgränsnitt
ping	Skickar echo-request paket och rapporterar hur lång tid det tar att få svar
ping6	Version av **ping** för IPv6 nätverk
talk	Används för att chatta med en annan användare
telnet	Gränssnitt för TELNET protokollet
tftp	Trivial file transfer program
traceroute	Spårar vägen dina paket tar från värden du arbetar på till värden på ett annat nätverk. Visar alla hopp(gateways) som sker däremellan.

6.41. Perl-5.30.1

Paketet Perl innehåller Practical Extraction and Report Language.

Förväntad byggtid: 9.2 SBU
Nödvändigt disk-utrymme: 272 MB

6.41.1. Installation av Perl

Skapa först en grundläggande /etc /hosts fil, vilken kommer refereras till i en av Perls konfigurationsfiler samt i den valbara testsviten:

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

Denna version av Perl bygger nu Compress::Raw::Zlib och Compress::Raw::BZip2 modules. Som standard kommer Perl använda en intern kopia av källkoderna för byggnationen. Skicka följande kommando så att Perl kommer använda biblioteken vilka är installerade på systemet:

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

För att få full kontroll över hur Perl skapas, kan du ta bort alternativet “-des” från nedan kommando, och själv välja på vilket sätt som paketet ska byggas. Alternativt, använd kommandot exakt som nedan för att använda de standarder som Perl per automatik upptäcker:

```
sh Configure -des -Dprefix=/usr                 \
                  -Dvendorprefix=/usr           \
                  -Dman1dir=/usr/share/man/man1 \
                  -Dman3dir=/usr/share/man/man3 \
                  -Dpager="/usr/bin/less -isR"  \
                  -Duseshrplib                  \
                  -Dusethreads
```

Vad alternativen betyder:

`-Dvendorprefix=/usr`
Säkerställer att **Perl** vet hur det berättar för paket var de skall installera sina perl-moduler

`-Dpager="/usr/bin/less -isR"`
Säkerställer att **`less`** Används istället för **`more`**

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`
Eftersom Groff inte är installerat ännu, tror **Configure** att vi inte vill ha man pages för Perl. Dessa alternativen skriver över detta beslut.

`-Duseshrplib`
Bygg ett delat libperl vilket krävs av vissa perl moduler.

`-Dusethreads`
Bygg perl med stöd för trådar.

Kompilera:

```
make
```

Kör test (ca 11 SBU):

```
make test
```

Installera och städa upp:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

6.41.2. Innehåll Perl

Installerade program:	corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.30.1 (hård länk till perl), perlbug, perldoc, perlivp, perlthanks (hård länk till perlbug), piconv, pl2pm, pod2html, pod2man, pod2text, pod2usage, podchecker, podselect, prove, ptar, ptardiff, ptargrep, shasum, splain, xsubpp, och zipdetails
Installerade bibliotek:	Fler än som kan listas här
Installerade mappar:	/usr/lib/perl5

Korta beskrivningar

corelist	Kommandorads frontend för Module::CoreList
cpan	Interagera med Comprehensive Perl Archive Network (CPAN) från kommandorad
enc2xs	Bygger en Perl-förlängning för Encode modulen, från antingen Unicode Character Mappings eller Tel Encoding Files
encguess	Gissar encoding type av en eller fler filer
h2ph	Omvandla `.h` C header filer till `.ph` Perl header filer
h2xs	Omvandla `.h` C header filer till Perl extensions
instmodsh	Skalskript för att undersöka installerade Perl-moduler. Kan skapa en tarball från installerade moduler
json_pp	Konverterar data mellan vissa input och output format
libnetcfg	Kan användas för att konfigurera `libnet` Perl modulen
perl	Kombinerar vissa av de bästa funktionerna i C, **sed**, **awk** och **sh** i ett swiss-army-knife språk
perl5.30.1	En hård länk till **perl**
perlbug	Används för att genera bugg-rapporter om Perl, eller moduler som kommer med det, samt maila dem
perldoc	Visar dokumentation i pod-format vilket är inkluderat i Perls installationsträd eller i ett perl-skript
perlivp	Perls installations verifikations procedur. Kan användas för att säkerställa att Perl och dess bibliotek har Installerats korrekt.
perlthanks	Används för att genera “thank you”-meddelanden att maila till Perls utvecklare.
piconv	En Perl-version av character encoding converter **iconv**
pl2pm	Verktyg för att konvertera Perl4 .p1 filer till Perl5 .pm moduler
pod2html	Konverterar filer från pod format till HTML format
pod2man	Konverterar pod data till formaterad *roff input
pod2text	Konverterar pod data till formaterad ASCII text

pod2usage	Skriver användande-meddelanden från filers inbäddade pod docs
podchecker	Kollar syntax för pod format dokumentations filer
podselect	Visar specifierade sektioner av pod dokumentation
prove	Kommandorads verktyg för att köra test gentemot Test::Harness modulen
ptar	Ett **tar**-liknande program skrivet i Perl
ptardiff	Ett Perl program vilket jämför ett uppackat arkiv med ett ouppackat sådant
ptargrep	Ett Perl program vilket applicerar mönster-matching till innehållt av filer i ett tar-akriv
shasum	Skriver ut eller kollar SHA checksums
splain	Används för att tvinga fram varningsdiagnostik i Perl
xsubpp	Konverterar Perl XS kod till C kod
zipdetails	Visar detaljer om den interna strukturen hos en Zip-fil

6.42. XML::Parser-2.46

Modulen XML::Parser är ett Perl gränssnitt för James Clark's XML parser, Expat.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 2.4 MB

6.42.1. Installation av XML::Parser

Förbered för kompilering:

```
perl Makefile.PL
```

Kompilera:

```
make
```

Kör test:

```
make test
```

Installera :

```
make install
```

6.42.2. Innehåll XML::Parser

Installerade moduler: Expat.so

Korta beskrivningar

`Expat` Tillhandahåller Perls Expat gränssnitt

6.43. Intltool-0.51.0

Intltool är ett internationaliserings verktyg vilket används för att extrahera översättbara strängar från källkodsfiler.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 1.5 MB

6.43.1. Installation av Intltool

Fixa först en varning som sker med perl-5.22 och senare:

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

Förbered för kompilering:

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

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera :

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

6.43.2. Innehåll Intltool

Installerade program: intltool-extract, intltool-merge, intltool-prepare, intltool-update, och intltoolize
Installerade mappar: /usr/share/doc/intltool-0.51.0 och /usr/share/intltool

Korta beskrivningar

intltoolize Förbereder ett paket för att använda intltool

intltool-extract Genererar headerfiler som kan läsas av **gettext**

intltool-merge Sammanfogar översatta strängar, till diverse olika filtyper

intltool-prepare Uppdaterar pot-filer och sammanfogar dem med översättningsfiler

intltool-update Uppdaterar pot-mallfiler och sammanfogar dem med översättningarna

6.44. Autoconf-2.69

Paketet Autoconf innehåller program för att producera skalskript som automatiskt kan konfigurera källkod.

Förväntad byggtid: Mindre än 0.1 SBU (3.2 med test)
Nödvändigt disk-utrymme: 79 MB

6.44.1. Installation av Autoconf

Fixa först en bugg genererad av Perl 5.28.

```
sed '361 s/{/\\{/' -i bin/autoscan.in
```

Förbered för kompilering:

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

Kompilera:

```
make
```

Testsviten är för tillfället trasig pga bash-5 och libtool-2.4.3. För att köra testen ändå:

```
make check
```

Installera :

```
make install
```

6.44.2. Innehåll Autoconf

Installerade program: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, och ifnames
Installerade mappar: /usr/share/autoconf

Korta beskrivningar

autoconf Producera skalskript som automatiskt konfigurerar källkods paket för att få dem att anpassa sig till många olika typer av Unix-lika system. Skripten vilka produceras är självständiga, och behöver inte autoconf för att köras.

autoheader Verktyg för att skapa mallfiler av C *#define* statements för configure att använda

autom4te En wrapper för makro processorn M4

autoreconf Automatiskt kör **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize**, och **libtoolize** i den korrekta ordningen för att spara tid då förändringar görs av **autoconf** och **automake** mallfiler.

autoscan Hjälper till i skapandet av configure.in filer för mjukvarupaket. Undersöker källkodsfilerna i ett mappträd, söker igenom dem för portabilitets-problem, och skapar en configure.scan fil vilken tjänar som en preliminär configure.in för paketet.

autoupdate Modifierar en configure.in, vilken fortfarande kallar **autoconf** makros med deras gamla namn, så att den använder dessa makros nuvarande namn.

ifnames Hjälper till i skapandet av configure.in för ett mjukvarupaket. Skriver ut identifierarna som paketet använder i C preprocessor conditionals. Om ett paket redan har definierats till att ha en viss mån av

portabilitet, kan detta program hjälpa till med att avgöra vad **configure** behöver kolla efter. Det kan också fylla i tomrum i en configure.in som genererats av **autoscan**.

6.45. Automake-1.16.1

Paketet Automake innehåller paket för att genera Makefiles för användning med Autoconf.

Förväntad byggtid: Mindre än 0.1 SBU (8.1 SBU med test)
Nödvändigt disk-utrymme: 107 MB

6.45.1. Installation av Automake

Förbered för kompilering:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.16.1
```

Kompilera:

```
make
```

Genom att använda -j4 make alternativet snabbas testen upp, även på system med bara en processor, på grund av interna fördröjningar i individuella test. Kör testen:

```
make -j4 check
```

Ett test brukar fallera: subobj.sh.

Installera:

```
make install
```

6.45.2. Innehåll Automake

Installerade program: aclocal, aclocal-1.16 (hårdlänkad med aclocal), automake, och automake-1.16 (hårdlänkad med automake)
Installerade mappar: /usr/share/aclocal-1.16, /usr/share/automake-1.16, och /usr/share/doc/automake-1.16.1

Korta beskrivningar

aclocal Genererar `aclocal.m4` filer baserade på innehållet i `configure.in` filer

aclocal-1.16 En hård länk till **aclocal**

automake Ett verktyg för att automatiskt generera Makefile.in filer från Makefile.am filer. För att skapa alla Makefile.in filer för ett paket, kör detta program i den översta nivån av mappträdet. Genom att skanna configure.in, hittar det automatiskt varje lämplig Makefile.am och generar den motsvarande Makefile.in.

automake-1.16 En hård länk till **automake**

6.46. Kmod-26

Paketet Kmod innehåller bibliotek och verktyg för att ladda kernelmoduler

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 13 MB

6.46.1. Installation av Kmod

Förbered för kompilering:

```
./configure --prefix=/usr          \
            --bindir=/bin          \
            --sysconfdir=/etc      \
            --with-rootlibdir=/lib \
            --with-xz              \
            --with-zlib
```

Vad alternativen betyder:

`--with-xz, --with-zlib`
Möjliggör för Kmod att hantera komprimerade kernelmoduler

`--with-rootlibdir=/lib`
Säkerställer att diverse biblioteksrelaterade filer placeras i de korrekta mapparna

Kompilera:

```
make
```

Detta paket inkluderar inte en testsvit som som kan köras i LFS chroot miljö. Som minimum krävs Git, och flertalet test vägrar köras utanför ett Git-repo.

Installera paketet, och skapa symlänkar för kompabilitet med Module-Init-Tools (paketet som tidigare hanterade Linux kernel moduler):

```
make install

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

ln -sfv kmod /bin/lsmod
```

6.46.2. Innehåll Kmod

Installerade program: depmod (länk till kmod), insmod (länk till kmod), kmod, lsmod (länk till kmod), modinfo (länk till kmod), modprobe (länk till kmod), och rmmod (länk till kmod)
Installerade bibliotek: libkmod.so

Korta beskrivningar

depmod Skapar en beroende(dependency)fil baserad på symbolerna det hittar i den existerande samlingen av moduler. Denna beroendefil används av **modprobe** för att automatiskt ladda de krävda modulerna.

insmod Installerar i kerneln en laddbar modul

kmod Laddar och avladdar kernelmoduler.

lsmod Listar nuvarande laddade moduler

modinfo Undersöker en objektfil relaterad med en kernelmodul och visar den information som kunde upptäckas

modprobe Använder en beroendefil skapad av **depmod**, för att automatiskt ladda relevanta moduler

rmmod Avladdar moduler från den aktiva kerneln

`libkmod` Detta bibliotek används av andra program för att ladda och avladda kernelmoduler

6.47. Gettext-0.20.1

Paketet Gettext innehåller verktyg för internationalisering och lokalisering. Dessa tillåter program att kompileras med NLS(Native Language Support), vilket möjliggör för dem att skriva ut meddelanden i användarens modersmål.

Förväntad byggtid: 2.7 SBU
Nödvändigt disk-utrymme: 249 MB

6.47.1. Installation av Gettext

Förbered för kompilering:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.20.1
```

Kompilera:

```
make
```

Kör test (3 SBU):

```
make check
```

Installera :

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

6.47.2. Innehåll Gettext

Installerade program: autopoint, envsubst, gettext, gettext.sh, gettextize, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin, och xgettext
Installerade bibliotek: libasprintf.so, libgettextlib.so, libgettextpo.so, libgettextsrc.so, libtextstyle.so, och preloadable_libintl.so
Installerade mappar: /usr/lib/gettext, /usr/share/doc/gettext-0.20.1, /usr/share/gettext, och /usr/share/gettext-0.19.8

Korta beskrivningar

autopoint Kopierar standarad Gettext infrastruktur filer in till ett källkodspaket

envsubst Ersätter miljövariabler i skalformat strängar

gettext Översätter ett meddelande till användarens språk genom att kolla upp översättningen i en meddelandekatalog

gettext.sh Fungerar primärt som ett skalfunktions bibliotek för gettext

gettextize Kopierar alla standard Gettext filer till den givna översta-nivån mappen i ett paket, för att påbörja internationalisering av det

msgattrib Filtrerar en översättningskatalogs meddelanden enligt deras attribut och manipulerar dessa attribut

msgcat	Konkatenerar och sammanfogar de givna .po filerna
msgcmp	Jämför två .po filer för att kolla att båda innehåller samma set av msgid strängar
msgcomm	Innehåller meddelanden som är vanliga för de givna .po filerna.
msgconv	Omvandlar en översättningskatalog till en annan typ av karaktärs encoding
msgen	Skapar en Engelsk översättningskatalog
msgexec	Applicerar ett kommando till alla översättningar i en översättningskatalog
msgfilter	Applicerar ett filter till alla översättningar i en översättningskatalog
msgfmt	Genererar en binär meddelandekatalog från en översättningskatalog
msggrep	Extraherar alla meddelanden, vilka matchar ett givet mönster eller tillhör någon av ett antal givna källkodsfiler, i en översättningskatalog.
msginit	Skapar en ny .po fil, initialiserande meta-informationen med värden från användarens Miljö.
msgmerge	Kombinerar två råa översättningar till en enskild fil
msgunfmt	Dekompilerar en binär meddelandekatalog till en rå översättningstext
msguniq	Enar likadana översättningar i en översättningskatalog
ngettext	Visar modersmåls språköversättning av ett textmeddelande vars grammatiska form beror på ett nummer.
recode-sr-latin	Omkodar Serbisk text från Kyrylliska till Latinska skript
xgettext	Extraherar de översättbara meddelanderaderna från de givna källkodsfilerna för att skapa de första översättnings mallarna.
`libasprintf`	Definierar klassen autosprint, vilken gör så att C-formaterade output-rutiner kan användas i C++ program, för användning med <string> strängar och <iostream> strömmar.
`libgettextlib`	Ett privat bibliotek som innehåller vanliga rutiner använda av de diverse Gettext programmen. Dessa rutiner är inte tänkta för allmän användning
`libgettextpo`	Används för att skriva specialiserade program vilka processar .po filer. Detta bibliotek används då standardapplikationerna som kommer med Gettext (såsom **msgcomm, msgcmp, msgattrib, msgen**) inte är tillräckliga.
`libgettextsrc`	Ett privat bibliotek som innehåller vanliga rutiner vilka används av de diverse gettext Programmen. Dessa är inte tänkta att användas rent allmänt
`libtextstyle`	Text stils bibliotek
`preloadable_libintl`	Ett bibliotek, tänkt att användas av LD_PRELOAD som stöjder `libintl` när det kommer till att logga icke översatta meddelanden.

6.48. Libelf from Elfutils-0.178

Libelf är ett bibliotek för att hantera ELF (Executable and Linkable Format) filer.

Förväntad byggtid: 0.9 SBU
Nödvändigt disk-utrymme: 124 MB

6.48.1. Installation av Libelf

Libelf är del av elfutils-0.178 paketet. Använd elfutils-0.178.tar.bz2 som källkods tarball.

Förbered för kompilering:

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

Kompilera:

```
make
```

Kör test:

```
make check
```

Ett test, run-elfclassify.sh, brukar misslyckas.

Installera endast Libelf:

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

6.48.2. Innehåll Libelf

Installerade bibliotek: libelf.so
Installerade mappar: /usr/include/elfutils

6.49. Libffi-3.3

Biblioteket Libffi tillhandahåller ett portabelt högnivå programmerings gränssnitt för diverse kallande konventioner. Detta tillåter en programmerare att kalla funktioner specifierade av ett sådant gränsnitts beskrivning vid run time.

Förväntad byggtid: 1.9 SBU
Nödvändigt disk-utrymme: 10 MB

6.49.1. Installation av Libffi

Kommentar

Likt GMP, bygger libffi med optimisering specifika för den använda processorn. Om man bygger för ett annat system, bör flaggorna CFLAGS och CXXFLAGS exporteras för att specifiera en generisk byggnation. Om detta inte görs, kommer alla applikationer vilka länkar till libffi att trigga Illegal Operation Errors.

Förbered för kompilering:

```
./configure --prefix=/usr --disable-static --with-gcc-arch=native
```

Vad alternativen betyder:

`--with-gcc-arch=native`
Säkerställ att gcc optimiserar för det nuvarande systemet. Ifall detta inte specifieras, gissar kompileringen systemet och koden som genereras kanske inte är korrekt. Om den genererade koden ska kopieras från det ursprungliga systemet till ett mindre kapabelt sådant, använd det mindre kapabla systemet som parameter. För detaljer om alternativa systemtyper, se *the x86 options in the gcc manual*.

Kompilera:

```
make
```

Kör test:

```
make check
```

Sex stycken test, alla relaterade till test-callback.c, brukar misslyckas.

Installera :

```
make install
```

6.49.2. Innehåll Libffi

Installerade bibliotek: libffi.so

Korta beskrivningar

`libffi` Innehåller libffi API funktioner.

6.50. OpenSSL-1.1.1d

Paketet OpenSSL innehåller verktyg och bibliotek relaterade till kryptografi. Dessa är användbara för att tillhandahålla kryptografiska funktioner åt andra paket, såsom OpenSSH, email applikationer och webbläsare(t.ex https).

Förväntad byggtid: 2.1 SBU
Nödvändigt disk-utrymme: 146 MB

6.50.1. Installation av OpenSSL

Förbered för kompilering:

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

Kompilera:

```
make
```

Kör test:

```
make test
```

Ett test i 20-test_enc.t brukar misslyckas.

Installera:

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

Om önskat, installera dokumentationen:

```
mv -v /usr/share/doc/openssl /usr/share/doc/openssl-1.1.1d
cp -vfr doc/* /usr/share/doc/openssl-1.1.1d
```

6.50.2. Innehåll OpenSSL

Installerade program: c_rehash och openssl
Installerade bibliotek: libcrypto.{so,a} och libssl.{so,a}
Installerade mappar: /etc/ssl, /usr/include/openssl, /usr/lib/engines och /usr/share/doc/openssl-1.1.1d

Korta beskrivningar

c_rehash Ett perl skript som skannar alla filer i en mapp och adderar symboliska länkar till deras hashvärden

openssl Ett kommandorads verktyg för att använda de diverse funktionerna i OpenSSLs bibliotek. En översikt vad gäller dessa funktioner återfinns i **man 1 openssl**

`libcrypto.so` Implementerar en omfattande samling kryptografiska algoritmer som används i diverse internet-standarder. Tjänsterna tillhandahållna av detta bibliotek används av OpenSSL-implementationen av

SSL, TLS och S/MIME. De har även använts för att implementera OpenSSH, OpenPGP, samt diverse andra kryptografiska standarder.

`libssl.so` Implementerar Transport Layer Security (TLS v1) protokollet. Tillhandahåller ett omfattande API, vars dokumentation återfinns via **man 3 ssl**

6.51. Python-3.8.1

Paketet Python 3 innehåller Pythons utvecklingsmiljö. Användbart för objektorienterad programmering, att skriva skript, skapa prototyper för stora program, samt utveckla fullständiga applikationer.

Förväntad byggtid: 1.2 SBU
Nödvändigt disk-utrymme: 426 MB

6.51.1. Installation av Python 3

Förbered för kompilering:

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

Vad alternativen betyder:

`--with-system-expat`
Möjliggör länkning gentemot systemets version av Expat.

`--with-system-ffi`
Möjliggör länkning gentemot systemets version av Libffi.

`--with-ensurepip=yes`
Möjliggör byggandet av **pip** och **setuptools** paketerings-program.

Kompilera:

```
make
```

För att testa resultaten, kör **make test**. Vissa test som kräver nätverksuppkoppling eller ytterligare paket hoppas över. testen som kallas test_normalization misslyckas eftersom nätverkskonfiguration inte är slutförd ännu. För mer omfattande resultat, kan testen köras igen då Python 3 återinstalleras i BLFS.

Installera:

```
make install
chmod -v 755 /usr/lib/libpython3.8.so
chmod -v 755 /usr/lib/libpython3.so
ln -sfv pip3.8 /usr/bin/pip3
```

Vad alternativen betyder:

chmod -v 755 /usr/lib/libpython3.{8.,}so
Fixa rättigheter så att de är desamma som för andra bibliotek

Om önskas, installera dokumentationen:

```
install -v -dm755 /usr/share/doc/python-3.8.1/html

tar --strip-components=1  \
    --no-same-owner       \
    --no-same-permissions \
    -C /usr/share/doc/python-3.8.1/html \
    -xvf ../python-3.8.1-docs-html.tar.bz2
```

Vad alternativen betyder:

`--no-same-owner` och `--no-same-permissions`
Säkerställ att de installerade filerna har korrekt ägarskap och rättigheter. Utan dessa alternativen, kommer tar att installera paketfilerna med upstream skaparens värden.

6.51.2. Innehåll Python 3

Installerade program: 2to3, idle3, pip3, pydoc3, python3, och python3-config
Installerade bibliotek: libpython3.8.so och libpython3.so
Installerade mappar: /usr/include/python3.8, /usr/lib/python3, och /usr/share/doc/python-3.8.1

Korta beskrivningar

2to3 Är ett python-program som läser Python 2.x källlkod och applicerar en serie av fixar för att omvandla den till python 3.x källkod.

idle3 Är ett wrapper skript som öppnar en Python medveten GUI editor. För att detta skript ska gå att köra, måste du ha installerat Tk innan Python så att Tkinter python-modulen byggs.

pip3 Paketinstalleraren för python. Du kan använda Pip för att installera paket från Python Package Index och även andra index.

pydoc3 Pythons dokumentationsverktyg

python3 Är ett tolkat, interaktivt, objekt-orienterat programmeringsspråk.

6.52. Ninja-1.10.0

Ninja är ett byggsystem som fokuserar på hastighet.

Förväntad byggtid: 0.3 SBU
Nödvändigt disk-utrymme: 89 MB

6.52.1. Installation av Ninja

Då det körs, kör Ninja vanligtvis ett maximalt antal processer parallellt. Som standard är detta antalet kärnor på systemet plus 2. i vissa fall kan detta överhetta processorn eller leda till att systemet får slut på minne. Om Ninja körs från kommandoraden, kan man skicka en -jN parameter för att begränsa antalet parallella processer, men vissa paket bäddar in exekveringen av Ninja och bifogar ingen -j parameter.

Genom att använda den alternative proceduren nedan blir det möjligt för en användare att begränsa antalet parallella processer via en miljövariabel NINJAJOBS. Till exempel kan den sättas till 4:

```
export NINJAJOBS=4
```

Vilket begränsar Ninja till 4 parallella processer.

Om önskat, lägg till möjligheten att använda NINJAJOBS genom att köra:

```
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
```

Kompilera:

```
python3 configure.py --bootstrap
```

Vad alternativen betyder:

`--bootstrap`
Tvingar Ninja att bygga om sig självt för det nuvarande systemet

Kör test:

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

Installera:

```
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
```

6.52.2. Innehåll Ninja

Installerade program: ninja

Korta beskrivningar

ninja Själva Ninja byggsystemet.

6.53. Meson-0.53.1

Meson är ett byggsystem med öppen källkod menat att vara både extremt snabbt samt så användarvänligt som möjligt.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 31 MB

6.53.1. Installation av Meson

Kompilera:

```
python3 setup.py build
```

Detta paket kommer inte med en testsvit.

Installera:

```
python3 setup.py install --root=dest
cp -rv dest/* /
```

Vad alternativen betyder:

`--root=dest`
Som standard installerar **python3 setup.py install** diverse file (såsom man pages) in i Python Eggs. Med en specifik root plats, installerar **setup.py** dessa filer i en standard hierarki. Då kan vi helt enkelt bara kopiera hierarkin så att filerna hamnar i standard platsen.

6.53.2. Innehåll Meson

Installerade program: meson
Installerade mappar: /usr/lib/python3.8/site-packages/meson-0.53.1-py3.8.egg-info och /usr/lib/python3.8/site-packages/mesonbuild

Korta beskrivningar

meson Ett byggsystem som fokuserar på hög produktivitet

6.54. Coreutils-8.31

Paketet Coreutils innehåller verktyg för att visa och definiera systemets grundläggande karaktär.

Förväntad byggtid: 2.3 SBU
Nödvändigt disk-utrymme: 202 MB

6.54.1. Installation av Coreutils

POSIX kräver att program från Coreutils känner igen character boundaries korrekt även för multibyte locales. Följande patch fixar detta ickestöd och andra internationaliserings relaterade buggar:

```
patch -Np1 -i ../coreutils-8.31-i18n-1.patch
```

Kommentar

Tidigare återfanns det många buggar i denna patch. Då du rapporterar in buggar till Coreutils utvecklare, var vänlig att först kolla om de kan reproduceras utan denna patch.

Undvik ett test vilket på vissa maskiner kan loopa förevigt:

```
sed -i '/test.lock/s/^/#/' gnulib-tests/gnulib.mk
```

Förbered för kompilering:

```
autoreconf -fiv
FORCE_UNSAFE_CONFIGURE=1 ./configure \
            --prefix=/usr            \
            --enable-no-install-program=kill,uptime
```

Vad alternativen betyder:

autoreconf
Uppdaterar genererade konfigurationsfiler så de blir kompatbila med senaste versionen av automake

`FORCE_UNSAFE_CONFIGURE=1`
Denna miljövariabel tillåter paketet att byggas med användare root

`--enable-no-install-program=kill,uptime`
Förhindrar Coreutils från att installera binärer vilka senare kommer att installeras av andra paket.

Kompilera:

```
make
```

Nu är testsviten redo att köras. Kör först testen vilka är menade att köra som användare `root`:

```
make NON_ROOT_USERNAME=nobody check-root
```

Vi kommer att köra resten av testen som användaren `nobody`. Vissa test kräven däremot att användare är medlem av mer än en grupp. Vi skapar därför en temporär grupp och gör så att användaren “nobody” är medlem i den:

```
echo "dummy:x:1000:nobody" >> /etc/group
```

Fixa vissa av rättigheterna så att även icke-root användaren kan kompilera och köra testen:

```
chown -Rv nobody .
```

Kör nu testen. Säkerställ att PATH i miljön **su** inkluderar /tools /bin.

```
su nobody -s /bin/bash \
          -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes check"
```

Testprogrammet test-getlogin kan misslyckas i en delvis byggd systemmiljö, likt chroot miljön här, men lyckas ifall det körs i slutet av detta kapitel. Testprogrammet tty.sh kan också misslyckas.

Radera den temporära gruppen:

```
sed -i '/dummy/d' /etc/group
```

Installera:

```
make install
```

Flytta program till destionationerna specifierade av FHS:

```
mv -v /usr/bin/{cat,chgrp,chmod,chown,cp,date,dd,df,echo} /bin
mv -v /usr/bin/{false,ln,ls,mkdir,mknod,mv,pwd,rm} /bin
mv -v /usr/bin/{rmdir,stty,sync,true,uname} /bin
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\"/1 /usr/share/man/man8/chroot.8
```

Vissa av skripten i paketet LFS-Bootscripts är beroende av **head**, **nice**, **sleep**, and **touch**. Då /usr kanske inte är tillgänglig under de tidiga och sena stadierna av uppstart, behöver dessa binärer återfinnas på root-partitionen för att efterleva FHS kravställning.

```
mv -v /usr/bin/{head,nice,sleep,touch} /bin
```

6.54.2. Innehåll Coreutils

Installerade program: [, 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, och yes
Installerade bibliotek: libstdbuf.so (in /usr/libexec/coreutils)
Installerade mappar: /usr/libexec/coreutils

Korta beskrivningar

base32 Encodes och decodes data enligt base32 specifikationen (RFC 4648)

base64 Encodes och decodes data enligt base64 specifikationen (RFC 4648)

b2sum Skriver eller dubbelkollar BLAKE2 (512-bit) checksums

basename Tar bort sökväg och suffix från ett filnamn

basenc Encodes eller decodes data med hjälp av diverse algoritmer

cat Konkatenerar filer till standard output

chcon Ändrar säkerhetskontext för filer och mappar.

chgrp Ändrar gruppägarskap för filer och mappar.

chmod Ändrar rättigheter för varje fil till det specifierade tillståndet. Tillståndet kan antingen vara en symbolisk representation av förändringarna som skall göras eller ett octal-nummer representerande de nya rättigheter

chown Ändrar användare och/eller grupp ägarskapet av filer och mappar

chroot Kör ett kommando med den specifierade mappen som / mapp

cksum Skriver ut Cyclic Redundancy Check (CRC) checksum och byte-antal för varje specifierad fil

comm Jämför två sorterade filer, skrivande ut i tre kolumner de rader som är unika samt de som är vanliga

cp Kopierar filer

csplit Delar upp en given fill i flertalet nya, separerande dem enligt givna mönster och radnummer, och skriver ut byte-antal för varje ny fil

cut Skriver ut sektioner av rader, där rader väljs med hjälp av givna fält och positioner

date Visar den nuvarande tiden i ett givet format, eller definierar systemets datum

dd Kopierar en fil enligt den givna block-storleken. Utför som alternativ även omvandlingar på den.

df Rapporterar mängd diskutrymme som finns tillgängligt (och används) på alla monterade filsystem, eller endast på systemen som lagrar de valda filerna

dir Listar innehållet i varje given mapp (samma som **ls**)

dircolors Skriver ut kommandon så LS_COLOR miljövariabeln ändrar färgschemat som används av **ls**

dirname Raderar icke-mapp-suffix från ett filnamn

du Rapporterar mängden av diskutrymme som används av den nuvarande mappen, av specifierade mappar eller av specifierade filer

echo Visar de specifierade strängarna

env Kör ett kommando i en modifierad miljö

expand Omvandlar tabs till mellanslag

expr Utvärderar uttryck

factor Skriver ut primfaktorerna av alla specifierade integernummer

false Gör inget, men misslyckas; exitar alltid med en statuskod som indikerar misslyckande

fmt Omformaterar paragrafer i de givna filerna

fold Wrappar raderna i de givna filerna

groups Rapporterar en användares gruppmedlemskap

head Skriver de första tio raderna (eller det givna antalet av rader) för varje given fil

hostid Rapporterar den numeriska identifieraren (i hexadecimal) för värden

id	Rapporterar UID, GID, samt gruppmedlemskap för den nuvarande eller specifierade användaren
install	Kopier filer medan definierande deras rättighets-tillstånd och, om möjligt, deras ägare och grupp
join	Sammanfogar raderna, som har identiska join fields, från två filer
link	Skapar en hård länk med det givna namnet för en fil
ln	Skapa hårda eller mjuka(symboliska) länkar mellan filer
logname	Rapporterar den nuvarande användarens användarnamn
ls	Listar innehållet av varje given mapp
md5sum	Rapporterar eller kollar Message Digest 5 (MD5) checksums
mkdir	Skapar mappar med givet namn eller givna namn
mkfifo	Skapar First-In, First-Outs (FIFOs), en "named pipe" i UNIX slang, med de givna namnen
mknod	Skapar enhetsnoder med de givna namnen. En enhetsnod är en character special fil, en block special fil, eller en FIFO.
mktemp	Skapar temporära filer på ett säkert vis. Används i skript.
mv	Flyttar eller byter namn på filer eller mappar
nice	Kör ett program med modifierad schematisk prioritet
nl	Numrerar raderna för den givna filen
nohup	Kör ett kommando som är immunt mot att hänga sig, vars output omdirigeras till en logfil
nproc	Skriver ut antalet processerings enheter som finns tillgängliga för en process
numfmt	Omvandlar nummer till eller från människo-läsbara strängar
od	Dumpar filer i octal eller andra format
paste	Sammanfogar de givna filerna
pathchk	Dubbelkollar ifall filnamn är korrekta eller portabla
pinky	Lättvikts finger klient. Rapporterar information om den specifierade användaren
pr	Paginates och columnates filer för att skrivas
printenv	Skriver miljön
printf	Skriver de givna argumenten enligt det givna formatet. Mycket likt Cs funktion printf
ptx	Producerar ett permuted index från innehåller av de givna filerna, med varje nyckelord i sin kontext
pwd	Rapporterar namnet för den nuvarande mappen
readlink	Rapporterar värdet av den givna symboliska länken
realpath	Skriver den resolved path
rm	Raderar filer eller mappar
rmdir	Raderar mappar ifall de är tomma
runcon	Kör ett kommando med specifierad säkerhetskontext
seq	Skriver en sekvens av nummer inom en given begränsning och med ett givet inkrement
sha1sum	Skriver eller dubbelkollar 160-bit Secure Hash Algorithm 1 (SHA1) checksums
sha224sum	Skriver eller dubbelkollar 224-bit Secure Hash Algorithm checksums

sha256sum	Skriver eller dubbelkollar 256-bit Secure Hash Algorithm checksums
sha384sum	Skriver eller dubbelkollar 384-bit Secure Hash Algorithm checksums
sha512sum	Skriver eller dubbelkollar 512-bit Secure Hash Algorithm checksums
shred	Skriver upprepade gånger över de givna filerna med komplexa mönster. Gör det svårt att återskapa filerna
shuf	Blandar rader av text
sleep	Pausar under den givna tidsrymden
sort	Sorterar raderna för de givna filerna
split	Delar upp den givna filen, enligt storlek eller antal rader
stat	Visar fil eller filsystem status
stdbuf	Kör kommandon med för dess standard streams förändrade buffering-operationer
stty	Definierar eller rapporterar terminalens radinställningar
sum	Skriver checksum och block counts för varje given fil
sync	Rensar filsystem buffers. Tvingar förändrade block till disken och uppdaterar super-block
tac	Konkatenerar de givna filerna baklänges
tail	Skriver de senaste 10 raderna(eller specifierade antal raderna) av varje given fil
tee	Läser från standard input medan skrivande både till standard output och de givna filerna
test	Jämför värden och dubbelkollar filtyper
timeout	Kör ett kommando med en tidsgräns
touch	Ändrar filers tidsstämplar, definierande tillgång och modifierings tider för de givna filerna till den nuvarande tiden och datum. Filer som inte existerar skapas med 0 i längd.
tr	Översätter, squeezes, och raderar de givna karaktärerna från standard input
true	Gör inget, framgångsrikt. Exitar alltid med en statuskod som indikerar framgång
truncate	Krymper eller expanderar en fil till en specifierad storlek
tsort	Genomför en topologisk sortering. it writes a completely ordered list according to the partial ordering in a given file
tty	Rapporterar filnamnet för den terminal som är kopplad till standard input
uname	Rapporterar systeminformation
unexpand	Omvandlar mellanslag till tabs
uniq	Slänger alla utom en av successiva identiska rader
unlink	Raderar den givna filen
users	Rapporterar namnet för de användare som för tillfället är inloggad
vdir	Samma som **ls -l**
wc	Rapporterar antalet rader, ord, och bytes för varje given fil, och även en total radräkning då mer än en fil har givits till kommandot
who	Rapporterar vem som är inloggad
whoami	Rapporterar användarnamnet som är associerat med det nuvarande UID
yes	Outputs upprepade gånger -y (eller annan given sträng) tills dess att kommandot är dödat

`libstdbuf` Bibliotek som används av **stdbuf**

6.55. Check-0.14.0

Check är ett testnings-ramverk för C.

Förväntad byggtid: 0.1 SBU (3.5 SBU med test)
Nödvändigt disk-utrymme: 13 MB

6.55.1. Installation av Check

Förbered för kompilering:

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

Kompilera:

```
make
```

Kör test:

```
make check
```

Testsviten kan ta upp till 4 SBU.

Installera paketet och fixa ett skript:

```
make docdir=/usr/share/doc/check-0.14.0 install &&
sed -i '1 s/tools/usr/' /usr/bin/checkmk
```

6.55.2. Innehåll Check

Installerade program: checkmk
Installerade bibliotek: libcheck.{a,so}

Korta beskrivningar

checkmk Awk skript för att generera C unit test, för användning med Checks unit testing ramverk

`libcheck.{a,so}` Innehåller funktioner som tillåter Check att kallas från ett testprogram

6.56. Diffutils-3.7

Paketet Diffutils innehåller program som visar skillnaderna mellan mappar och filer.

Förväntad byggtid: 0.4 SBU
Nödvändigt disk-utrymme: 36 MB

6.56.1. Installation av Diffutils

Förbered för kompilering:

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

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

6.56.2. Innehåll Diffutils

Installerade program: cmp, diff, diff3, och sdiff

Korta beskrivningar

cmp Jämför två filer och rapporterar hur deras bytes skiljer sig åt

diff Jämför två filer eller mappar och rapporterar vilka rader i filerna som skiljer sig åt

diff3 Jämför tre filer rad för rad

sdiff Sammanfogar två filer och skriver interaktivt ut resultatet

6.57. Gawk-5.0.1

Paketet Gawk innehåller program för att manipulera textfiler.

Förväntad byggtid: 0.4 SBU
Nödvändigt disk-utrymme: 47 MB

6.57.1. Installation av Gawk

Säkerställ först att ett antal filer vilka inte behövs, inte installeras:

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

Förbered för kompilering:

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

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

Om önskat, installera dokumentationen:

```
mkdir -v /usr/share/doc/gawk-5.0.1
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-5.0.1
```

6.57.2. Innehåll Gawk

Installerade program: awk (länk till gawk), gawk, och awk-5.0.1
Installerade bibliotek: filefuncs.so, fnmatch.so, fork.so, inplace.so, intdiv.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoway.so, rwarray.so, och time.so (alla i /usr/lib/gawk)
Installerade mappar: /usr/lib/gawk, /usr/libexec/awk, /usr/share/awk, och /usr/share/doc/gawk-5.0.1

Korta beskrivningar

awk En länk till **gawk**

gawk Ett program för att manipulera textfiler; GNU implementationen av **awk**

gawk-5.0.1 En hård länk till **gawk**

6.58. Findutils-4.7.0

Paketet Findutils innehåller program för att hitta filer. Dessa program är tillhandahållna för att rekursivt söka igenom en mappstruktur och skapa, underhålla, samt genomsöka en databas (ofta snabbare än den rekursiva sökningen, men opålitlig ifall databasen inte uppdaterats nyligen).

Förväntad byggtid: 0.7 SBU
Nödvändigt disk-utrymme: 57 MB

6.58.1. Installation av Findutils

Förbered för kompilering:

```
./configure --prefix=/usr --localstatedir=/var/lib/locate
```

Vad alternativen betyder:

`--localstatedir`
Ändrar destinationen för **locate** databasen till att återfinnas i `/var/llocate`, vilken lever upp till FHS krav.

Kompilera:

```
make
```

Kör test:

```
make check
```

Två test misslyckas ofta i chroot miljön: sv-bug-54171.old-O3 och sv-bug-54171.new-O3.

Installera:

```
make install
```

Vissa av skripten i LFS-Bootscripts package är beroende av **find**. Eftersom /usr kanske inte finns tillgänglig under de tidiga stadierna av uppstart, behöver detta program återfinnas i root-partitionen. Skriptet för **updatedb** behöver också modifieras för att korrigera en specifik sökväg:

```
mv -v /usr/bin/find /bin
sed -i 's|find:=${BINDIR}|find:=/bin|' /usr/bin/updatedb
```

6.58.2. Innehåll Findutils

Installerade program: find, locate, updatedb, och xargs
Installerade mappar: /var/lib/locate

Korta beskrivningar

find Söker igenom givna mapp-träd för filer som matchar specifierade kriterier

locate Söker igenom en databas av filnamn och rapporterar namnen som innehåller en given sträng eller matchar ett givet mönster

updatedb Uppdaterar **locate** databasen; skannar hela filsystemet (inkluderat andra filsystem vilka för tillfället är monterade, ifall inte instruerad att inte göra detta) och placerar i databasen varje filnamn det hittar

xargs Kan användas för att applicera ett givet kommando på en lista av filer

6.59. Groff-1.22.4

Paketet Groff innehåller program för att processa och formatera text.

Förväntad byggtid: 0.5 SBU
Nödvändigt disk-utrymme: 95 MB

6.59.1. Installation av Groff

Groff förväntar sig att miljövariabeln PAGE innehåller standard pappersstorleken. För användare i USA är PAGE=LETTER lämpligt. För övriga platser är PAGE=A4 ofta mer lämpligt. Även om standard pappersstorlek konfigureras vid kompilering, kan den skrivas över genom modifiering av filen /etc /papersize.

Förbered för kompilering:

```
PAGE=<paper_size> ./configure --prefix=/usr
```

Detta paket har inte stöd för att byggas parallellt. Kompilera som en process:

```
make -j1
```

Detta paket inkluderar inte en testsvit.

Installera :

```
make install
```

6.59.2. Innehåll Groff

Installerade program: 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, och troff
Installerade mappar: /usr/lib/groff och /usr/share/doc/groff-1.22.4, /usr/share/groff

Korta beskrivningar

addftinfo Läser en troff fontfil och lägger till ytterligar font-metric information som används av **groff** systemet.

afmtodit Skapar en fontfil för användning av **groff** och **grops**

chem Groff preprocessor för att skapa kemiskstruktur-diagram

eqn Kompilerar beskrivningar av ekvationer inbäddade inuti troff inputfiler, till kommandon som kan bli förstådda av troff

eqn2graph Konverterar en troff EQN (ekvation) till en beskuren bild

gdiffmk Markerar skillnader emellan groff/nroff/troff filer

glilypond Omvandlar musik skriven i språket lilypond, till språket groff

gperl Preprocesor för groff, tillåter addering av perl kod till groff filer

gpinyin Preprocesor för groff, tillåter addering av språket Pinyin till groff filer.

grap2graph	Konverterar ett grap diagram till en beskuren bitmap bild
grn	En **groff** preprocessor för gremlin filer
grodvi	En drivrutin för **groff** som producerar TeX dvi format
groff	En frontend för groff dokument formaterings systemet. Vanligtvis kör det **troff** programmet och en preprocessor som är lämplig för den valda enheten
groffer	Visar groff filer och man pages på X och tty terminaler
grog	Läser filer och gissar vilka av alternativen för **groff** `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s`, och `-t` som behövs för att skriva ut filer, och rapporterar det **groff** kommando som innehåller dessa
grolbp	Är en **groff** drivrutin för Canon CAPSL skrivare (LBP-4 och LBP-8 laser skrivare)
grolj4	En **groff** drivrutin som producerar output PCL5 format lämpligt för en HP LaserJet 4 skrivare
gropdf	Översätter output från GNU **troff** till PDF
grops	Översätter output från GNU **troff** till PostScript
grotty	Översätter output från GNU **troff** till en form mer lämplig för typewriter-liknande enheter
hpftodit	Skapar från en HP-taggad font metric film, en font fil att används med **groff -Tlj4**
indxbib	Skapar ett omvänt index för de bibliografiska databaserna med en speciell fil för användning med **refer, lookbib, lkbib**
lkbib	Söker igenom bibliografiska databaser efter referenser som innehåller specifierade nycklar, och rapporterar vilka referenser som återfanns
lookbib	Skriver en prompt till standard error (bortsett från ifall standard input inte är en terminal), läser en rad som innehåller en samling av nyckelord från standard input, söker igenom de bibliografiska databaserna efter referenser som innehåller dessa nyckelord, skriver referenser vilka upptäcktes till standard output, och repeterar denna process tills input är slut
mmroff	En simpel preprocessor för **groff**
neqn	Formaterar ekvationer för American Standard Code for Information Interchange (ASCII) output
nroff	Ett skript som emulerar **nroff** kommandot genom att använda **groff**
pdfmom	En wrapper omkring groff vilken möjliggör skapandet av PDF dokument från filer formaterade med mom makron
pdfroff	Skapar PDF dokument med hjälp av groff
pfbtops	Översätter en PostScript font i `.pfb` format till ASCII
pic	Kompilerar beskrivningar av bilder inbäddad i troff eller TeX input filer till kommandon vilka kan bli förstådda av TeX eller **troff**
pic2graph	Konverterar ett PIC diagram till en beskuren bild
post-grohtml	Översätter output för GNU **troff** till HTML
preconv	Konverterar encoding av input filer till något vilket GNU **troff** förstår
pre-grohtml	Översätter GNU **troff** output till HTML
refer	Kopierar innehållet av en fil till standard output, bortsett från rader mellan .[och .] vilka tolkas som citat, samt rader mellan .R1 och .R2 vilka tolkas som kommandon för hur citat ska hanteras.
roff2dvi	Omvandlar roff filet till DVI format

roff2html	Omvandlar roff filer till HTML format
roff2pdf	Omvandlar roff filer till PDFer
roff2ps	Omvandlar roff filer till ps filer
roff2text	Omvandlar roff filer till text filer
roff2x	Omvandlar roff filer till andra format
soelim	Läser filer och ersätter rader av formen .so med innehållet hos den nämnda filen
tbl	Kompilerar beskrivningar av tables inbäddade i troff inputfiler, till kommandon vilka kan bli förstådda av **troff**
tfmtodit	Skapar en fontfil att användas med **groff -Tdvi**
troff	Kompatibel med Unix **troff**; bör vanligen kallas igenom kommandot **groff**, vilket även kommer köra preprocessors och postprocessors i den lämpliga ordningen och med de lämpliga alternativen

6.60. GRUB-2.04

Paketet GRUB innehåller GRand Unified Bootloader.

Förväntad byggtid: 0.8 SBU
Nödvändigt disk-utrymme: 161 MB

6.60.1. Installation av GRUB

Förbered för kompilering:

```
./configure --prefix=/usr          \
            --sbindir=/sbin        \
            --sysconfdir=/etc      \
            --disable-efiemu       \
            --disable-werror
```

Vad alternativen betyder:

`--disable-werror`
Tillåter byggnationen att slutföras även med varningar som introducerades med mer nyliga Flex versioner

`--disable-efiemu`
Minimiserar det som byggs genom att deaktivera en funktions samt testningsprogram som inte behövs i LFS

Kompilera:

```
make
```

Detta paket inkluderar inte en testsvit.

Installera:

```
make install
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

Att använda GRUB för att göra systemet uppstartsbart kommer diskuteras i Section 8.4, “Using GRUB to Set Up the Boot Process”.

6.60.2. Innehåll GRUB

Installerade program: 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, och grub-syslinux2cfg
Installerade mappar: /usr/lib/grub, /etc/grub.d, /usr/share/grub, och /boot/grub (då grub-install körs en första gång)

Korta beskrivningar

grub-bios-setup Är ett hjälpprogram för grub-install

grub-editenv Ett verktyg för att redigera miljö block

grub-file Dubbelkollar ifall FILE är av den specifierade typen.

grub-fstest	Verktyg för att felsöka filsystemets drivrutin
grub-glue-efi	Processar ia32 och amd64 EFI images and glues dem enligt formatet Apple.
grub-install	Installerar GRUB på din disk
grub-kbdcomp	Skript som konverterar xkb layout till en som känns igen av GRUB
grub-macbless	Mac-style bless på HFS eller HFS+ filer
grub-menulst2cfg	Omvandlar GRUB Legacy `menu.lst` till en `grub.cfg` att användas med GRUB 2
grub-mkconfig	Genererar en grub konfigurations fil
grub-mkimage	Skapar en uppstartsbar image av GRUB
grub-mklayout	Generar en GRUB tangentbords layout fil
grub-mknetdir	Förbered en GRUB netboot mapp
grub-mkpasswd-pbkdf2	Generar ett krypterat PBKDF2 lösenord för användning i bootmenyn
grub-mkrelpath	Gör ett system pathname relativt till dess root
grub-mkrescue	Skapar en uppstartsbar image lämplig för en diskett eller en CDROM/DVD
grub-mkstandalone	Skapar en självständig image
grub-ofpathname	Ett hjälpprogram som skriver sökvägen för en GRUB enhet
grub-probe	Söker igenom enhetsinformation för en given sökväg eller fil
grub-reboot	Definierar standard boot entry för GRUB, men endast för nästa uppstart
grub-render-label	Rendera Apple .disk_label för Apple Macs
grub-script-check	Dubbelkollar GRUB konfigurations skript efter syntax errors
grub-set-default	Definierar standard boot entry för GRUB
grub-sparc64-setup	Hjälpprogram för grub-setup
grub-syslinux2cfg	Omvandlar en syslinux config fil till formatet grub.cfg

6.61. Less-551

Paketet Less innehåller en textfils visare.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 4.1 MB

6.61.1. Installation av Less

Förbered för kompilering:

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

Vad alternativen betyder:

`--sysconfdir=/etc`
Instruerar programmen vilka skapas av paketet att leta i mappen `/etc` efter konfigurationsfilerna.

Kompilera:

```
make
```

Detta paket inkluderar inte en testsvit.

Installera:

```
make install
```

6.61.2. Innehåll Less

Installerade program: less, lessecho, och lesskey

Korta beskrivningar

less En filvisare. Den visar innehållet av en given fil, genom att tillåta användaren att scrolla igenom, söka ffter strängar, och hoppa till markeringar.

lessecho Behövs för att expandera meta-characters, såsom * och *?*, i filnamn på unix-system

lesskey Används för att specifiera key-bindings för **less**

6.62. Gzip-1.10

Paketet Gzip innehåller program för att komprimera och dekomprimera filer.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 20 MB

6.62.1. Installation av Gzip

Förbered för kompilering:

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

Kompilera:

```
make
```

Kör test:

```
make check
```

Två test misslyckas ofta: help-version and zmore.

Installera:

```
make install
```

Flytta ett program som behöver återfinnas på filsystemet root:

```
mv -v /usr/bin/gzip /bin
```

6.62.2. Innehåll Gzip

Installerade program: gunzip, gzexe, gzip, uncompress (hård länk med gunzip), zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore, och znew

Korta beskrivningar

gunzip Dekomprimerar gzippade filer

gzexe Skapar självdekomprimerande exekverbara filer

gzip Komprimerar de givna filerna med Lempel-Ziv (LZ77) kodning

uncompress Dekomprimerar komprimerade filer

zcat Dekomprimerar de givna gzippade filerna till standard output

zcmp Kör **cmp** på gzippade filer

zdiff Kör **diff** på gzippade filer

zegrep Kör **egrep** på gzippade filer

zfgrep Kör **fgrep** på gzippade filer

zforce Tving en .gz förlängning på alla givna filer som är gzippade, så att gzip inte komprimerar dem ännu en Gång. Kan vara användbart för filer vars namn förändrades vid exempelvis en filöverföring.

zgrep Kör **grep** på gzippade filer

zless	Kör **less** på gzippade filer
zmore	Kör **more** på gzippade filer
znew	Återkomprimerar filer från formatet **compress** till formatet **gzip** — .Z till -gz

6.63. Zstd-1.4.4

Zstandard är en realtids komprimerings algoritm, vilken tilhandahåller hög kompressionsratio. Den tillhandahåller ett stort antal kompressions/hastighets överväganden, samtidigt som den är understödd av en väldigt snabb decoder.

Förväntad byggtid: 0.7 SBU
Nödvändigt disk-utrymme: 16 MB

6.63.1. Installation av Zstd

Kompilera:

```
make
```

Detta paket inkluderar inte en testsvit.

Installera:

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

Radera det statiska biblioteket och flytta det delade biblioteket till /lib. Ytterligare behöver även .so filen i /usr /lib återskapas.

```
rm -v /usr/lib/libzstd.a
mv -v /usr/lib/libzstd.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libzstd.so) /usr/lib/libzstd.so
```

6.63.2. Innehåll Zstd

Installerade program: zstd, zstdcat (länk till zstd), zstdgrep, zstdless, zstdmt (länk till zstd), och unzstd (länk till zstd)
Installerade bibliotek: libzstd.so

Korta beskrivningar

zstd Komprimerar eller dekomprimerar filer med hjälp av formatet ZSTD

zstdgrep Kör **grep** på ZSTD komprimerade filer

zstdless Kör **less** på ZSTD komprimerade filer

`libzstd` Biblioteket vilket implementerar lossless data komprimering, genom att använda algoritmen ZSTD

6.64. IPRoute2-5.5.0

Paketet IPRoute2 innehåller program för grundläggande och avancerad ipv4-baserade nätverksfunktionalitet.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 14 MB

6.64.1. Installation av IPRoute2

Programmet **arpd** inkluderat i detta paket kommer inte byggas då det är beroende av Berkley DB, vilket inte är installerat i LFS. Däremot kommer en mapp samt en man page att installeras för **arpd**. Förhindra detta genom att köra kommandona nedan. Ifall **arpd** binären behövs, återfinns instruktioner för att kompilera Berkley DB i boken för BLFS.

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

Det är också nödvändigt att förhindra byggandet av två moduler som krävern *http://www.linuxfromscratch.org/blfs/view/9.1/postlfs/iptables.html*

```
sed -i 's/.m_ipt.o//' tc/Makefile
```

Kompilera:

```
make
```

Detta paket har inte en fungerande testsvit.

Installera:

```
make DOCDIR=/usr/share/doc/iproute2-5.5.0 install
```

6.64.2. Innehåll IPRoute2

Installerade program: bridge, ctstat (länk till lnstat), genl, ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (länk till lnstat), ss, och tc
Installerade mappar: /etc/iproute2, /usr/lib/tc, och /usr/share/doc/iproute2-5.5.0,

Korta beskrivningar

bridge Konfigurerar nätverksbroar

ctstat Uppkopplings status verktyg

genl Generiskt netlink verktygs frontend

ifcfg Ett shell script wrapper för kommandot **ip.** [Notera att det kräver programmen **arping** och **rdisk** från iputils paketet som återfinns vid http://www.skbuff.net/iputils/

ifstat Visar gränssnitts statistik, inkluderat mängden överförda samt mottagna paket per gränssnitt

ip Den huvudsakliga executabeln. Den har flertalet funktioner:
ip link *<device>* Tillåter användare att identifiera enheters tillstånd samt göra förändringar
ip addr Tillåter användare att identifiera adresser och deras värden, samt lägga till och radera adresser
ip neighbor Tillåter användare att identifiera neighbor bindings och deras värden, lägga till nya neighbor identifikationer samt radera gamla sådana

ip rule tillåter användare att kolla på routing policys samt ändra dem
ip route tillåter användare att kolla på routing table och ändra routing table regler
ip tunnel tillåter användare att kolla på IP tunnlar och deras egenskaper, samt ändra dem
ip maddr tillåter användare att kolla på multicast adresser och deras egenskaper, samt ändra dem
ip mroute tillåter användare att definera, ändra, eller radera multicast routingen
ip monitor tillåter användare att kontinuerligt monitorera tillståndet för enheter, adresser och routes

lnstat Tillhandahåller Linux nätverks statistik; en generaliserad och mer funktions-komplett ersättning för det tidigare använda programmet **rstat**

nstat Visar nätverks statistik

routef En **ip route** komponent. Den är till för att rensa routing tables

routel En **ip route** komponent. Den är till för att lista routing tables

rtacct Visar innehållet i filen `/proc/net/rt_acct`

rtmon Route monitorerings verktyg

rtpr Omvandlar output för **ip -o** tillbaka till läsbar form

rtstat Route status verktyg

ss Liknar kommandot **netstat**; visar aktiva uppkopplingar

tc Traffic Controlling Executable; den är till för Quality Of Service (QOS) och Class Of Service (COS) implementationer
tc qdisc tillåter användare att konfigurera queueing discipliner
tc class tillåter användare att konfigurera klasser basera på queuing discipline schemat
tc estimator tillåter användare att beräkna nätverksflödet till ett nätverk
tc filter tillåter användare att konfigurera QOS/COS paketfiltrering
tc policy tillåter användare att konfigurera QOS/COS policys

6.65. Kbd-2.2.0

Paketet Kbd innehåller key-table filer, konsol fonter, samt tangentbords verktyg.

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 36 MB

6.65.1. Installation of Kbd

Backspace och Delete tangenternas beteende är inte konsistent utöver keymaps i paketet Kbd. Följande patch fixar detta problem för i386 keymaps:

```
patch -Np1 -i ../kbd-2.2.0-backspace-1.patch
```

Efter patchning, genererar Backspace-tangenten karaktären som har kod 127, och Delete-tangenten genererar en välkänd escapesekvens.

Radera det ej nödvändiga programmet **resizecons** samt dess man pages:

```
sed -i 's/\(RESIZECONS_PROGS=\)yes/\1no/g' configure
sed -i 's/resizecons.8 //' docs/man/man8/Makefile.in
```

Förbered för kompilering:

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig ./configure --prefix=/usr --disable-vlock
```

Vad alternativen betyder

`--disable-vlock`
Förhindrar verktyget vlock från att byggas, eftersom det kräven biblioteket PAM, vilket inte är tillgängligt i chroot miljön.

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

Kommentar

För vissa språk (exempelvis Vitryska) tillhandahåller paketet Kbd inte en användbar keymap. Detta beror på skillnader i encoding, och använder av vissa språk behöver därför ladda ned önskade keymaps separat.

Om önskat, installera dokumentationen:

```
mkdir -v       /usr/share/doc/kbd-2.2.0
cp -R -v docs/doc/* /usr/share/doc/kbd-2.2.0
```

6.65.2. Innehåll Kbd

Installerade program: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (länk till psfxtable), psfgettable (länk till psfxtable), psfstriptable (länk till psfxtable), psfxtable, setfont, setkeycodes, setleds, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start, and unicode_stop

Installerade mappar: /usr/share/consolefonts, /usr/share/consoletrans, /usr/share/keymaps, /usr/share/doc/kbd-2.2.0, och /usr/share/unimaps

Korta beskrivningar

chvt Ändrar förgrundens virtuella terminal

deallocvt Deallokerar oanvända virtuella terminaler

dumpkeys Dumpar översättnings tables för tangentbordet

fgconsole Skriver antalet av aktiva virtuella terminaler

getkeycodes Skriver kernelns scancode-to-keycode mapping table

kbdinfo Erhåller information om en konsols status

kbd_mode Rapporterar eller definierar tangentbordets tillstånd

kbdrate Definierar tangentbordets upprepnings och fördröjnings inställningar

loadkeys Laddar översättnings tables för tangentbordet

loadunimap Laddar kernelns unicode-to-font mapping table

mapscrn Ett utgånget program som brukade ladda en användar-definierad character mapping table till konsolens drivrutin. Detta görs nu av **setfont**

openvt Startar ett program i en ny virtuell terminal (VT)

psfaddtable Adderar ett Unicode character table till en konsolfont

psfgettable Extraherar det inbäddade Unicode character table från en konsol font

psfstriptable Raderar det inbäddade Unicode character table från en konsol font

psfxtable Hanterar Unicode character tables för konsol fonts

setfont Ändrar Enhanced Graphic Adapter (EGA) och Video Graphics Array (VGA) fonterna för konsolen

setkeycodes Laddar kernel scancode-to-keycode mapping table entries; detta är användbart ifall det finns ovanliga knappar på tangentbordet

setleds Definierar tangentbords flaggor och Light Emitting Diodes (LEDs)

setmetamode Defnierar tangentbords meta-key handling

setvtrgb Definierar färg inställningar för alla virtuella terminalers konsoler

showconsolefont Visar den nuvarande EGA/VGA konsol skärms fonten

showkey Rapporterar scancodes, keycodes, och ASCII koder för knappar som trycks på tangentbordet

unicode_start Placerar tangentbord och konsol i UNICODE mode [använd inte detta program ifall inte din keymap fil är av ISO-8859-1 encoding. För andra encodings så producerar detta verktyg inkorrekta resultat.]

unicode_stop Återkallar tangentbord och konsol från UNICODE tillståndet

6.66. Libpipeline-1.5.2

Paketet Libpipeline innehåller ett paket för att manipulera pipelines hos subprocesser på ett flexibelt och enkelt vis.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 9.2 MB

6.66.1. Installation av Libpipeline

Förbered för kompilation:

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

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

6.66.2. Innehåll Libpipeline

Installerade bibliotek: libpipeline.so

Korta beskrivningar

`libpipeline` Bibliotek som används för att på ett säkert vis konstruera pipelines mellan subprocesser

6.67. Make-4.3

Paketet Make innehåller program för att kompilera paket.

Förväntad byggtid: 0.5 SBU
Nödvändigt disk-utrymme: 16 MB

6.67.1. Installation av Make

Förbered för kompilering:

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

Kompilera:

```
make
```

Testsviten behöver veta var understödjande perlfiler är lokaliserade. Vi använder en miljövariabel för detta syfte. för att testa resultaten, kör:

```
make PERL5LIB=$PWD/tests/ check
```

Installera:

```
make install
```

6.67.2. Innehåll Make

Installerade program: make

Korta beskrivningar

make Avgör per automatik vilka delar av ett paket som behöver (om)kompileras och utfärdar sedan de relevanta kommandona.

6.68. Patch-2.7.6

Paketet Patch innehåller ett program för att skapa och modifiera filer genom att applicera en “patch” fil vanligtvis skapad av programmet **diff**.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 13 MB

6.68.1. Installation av Patch

Förbered för kompilering:

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

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

6.68.2. Innehåll Patch

Installerade program: patch

Korta beskrivningar

patch Modifierar filer enligt en patchfil. En patchfil är vanligtvis en fil vilken listar skillnader, skapad av programmet **diff.** Genom att applicera dessa skillnader till originalfilen, skapar **patch** en patchad version av original-filen.

6.69. Man-DB-2.9.0

Paketet Man-DB innehåller program för att hitta och visa man pages.

Förväntad byggtid: 0.5 SBU
Nödvändigt disk-utrymme: 40 MB

6.69.1. Installation av Man-DB

Förbered för kompilering:

```
./configure --prefix=/usr                         \
            --docdir=/usr/share/doc/man-db-2.9.0 \
            --sysconfdir=/etc                     \
            --disable-setuid                      \
            --enable-cache-owner=bin              \
            --with-browser=/usr/bin/lynx          \
            --with-vgrind=/usr/bin/vgrind         \
            --with-grap=/usr/bin/grap             \
            --with-systemdtmpfilesdir=            \
            --with-systemdsystemunitdir=
```

Vad alternativen betyder:

`--disable-setuid`
Detta deaktiverar att setuid för **man** specifieras som användaren “man”

`--enable-cache-owner=bin`
Gör så att de systemövergripande cachefilerna ägs av användaren “bin”

`--with-...`
Dessa tre parameterar definierar ett antal standardprogram. **lynx** är en textbaserade webbläsare (se BLFS för instruktioner vad gäller installation). **Vgrind** konverterar programkällkod till Groff input, och **grap** är användbart för typesetting av grafer i groffdokument. Programmen **vgrind** och **grap** behövs vanligtvis inte för att visa manual sidor. De är inte del av LFS eller BLFS, men du bör ha möjlighet att installera dem ifall du önskar detta.

`--with-systemd...`
Förhindrar installation av oönskade systemd mappar och filer

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

6.69.2. Icke-engelska manualsidor för LFS

Nedan översikt visar karaktärs-seten som Man-DB antar att manualsidor installerade under /usr /share /man /<11> kommer vara encodeade med. Utöver detta, avgör Man-DB vare sig manualsidor installerade i denna mapp är UTF-8 encodeade.

Table 6.1. Förväntar karaktärs encoding för legacy 8-bit manualsidor

Language (code)	Encoding	Language (code)	Encoding
Danish (da)	ISO-8859-1	Croatian (hr)	ISO-8859-2
German (de)	ISO-8859-1	Hungarian (hu)	ISO-8859-2
English (en)	ISO-8859-1	Japanese (ja)	EUC-JP
Spanish (es)	ISO-8859-1	Korean (ko)	EUC-KR
Estonian (et)	ISO-8859-1	Lithuanian (lt)	ISO-8859-13
Finnish (fi)	ISO-8859-1	Latvian (lv)	ISO-8859-13
French (fr)	ISO-8859-1	Macedonian (mk)	ISO-8859-5
Irish (ga)	ISO-8859-1	Polish (pl)	ISO-8859-2
Galician (gl)	ISO-8859-1	Romanian (ro)	ISO-8859-2
Indonesian (id)	ISO-8859-1	Russian (ru)	KOI8-R
Icelandic (is)	ISO-8859-1	Slovak (sk)	ISO-8859-2
Italian (it)	ISO-8859-1	Slovenian (sl)	ISO-8859-2
Norwegian Bokmal (nb)	ISO-8859-1	Serbian Latin (sr@latin)	ISO-8859-2
Dutch (nl)	ISO-8859-1	Serbian (sr)	ISO-8859-5
Norwegian Nynorsk (nn)	ISO-8859-1	Turkish (tr)	ISO-8859-9
Norwegian (no)	ISO-8859-1	Ukrainian (uk)	KOI8-U
Portuguese (pt)	ISO-8859-1	Vietnamese (vi)	TCVN5712-1
Swedish (sv)	ISO-8859-1	Simplified Chinese (zh_CN)	GBK
Belarusian (be)	CP1251	Simplified Chinese, Singapore (zh_SG)	GBK
Bulgarian (bg)	CP1251	Traditional Chinese, Hong Kong (zh_HK)	BIG5HKSCS
Czech (cs)	ISO-8859-2	Traditional Chinese (zh_TW)	BIG5
Greek (el)	ISO-8859-7		

Kommentar

Manual sidor i språk som inte nämns i listan stödjs inte.

6.69.3. Innehåll Man-DB

Installerade program: accessdb, apropos (länk till whatis), catman, lexgrog, man, mandb, manpath, och whatis
Installerade bibliotek: libman.so och libmandb.so (båda i /usr/lib/man-db)
Installerade mappar: /usr/lib/man-db, /usr/libexec/man-db, och /usr/share/doc/man-db-2.9.0

Korta beskrivningar

accessdb Dumpar **whatis** databasens innehåll i människoläsbart format

apropos Söker igenom databasen **whatis** och visar de korta beskrivningarna för system kommandon vilka innehåller en given sträng

catman Skapar eller uppdaterar de för-formaterade manualsidorna

lexgrog Visar en-rads sammanfattande information om en given manualsida

man Formaterar och visar den efterfrågade manualsidan

mandb Skapar eller uppdaterar **whatis** databasen

manpath Visar innehållet för $MANPATH eller (ifall $MANPATH inte är definierad) en lämplig sökväg baserat på inställningarna i man.conf och användarens miljö

whatis Söker databasen **whatis** och visar de korta beskrivningarna för systemkommandon vilka innehåller ett givet nyckelord som ett separat ord

`libman` Innehåller run-time stöd för **man**

`libmandb` Innehåller run-time stöd för **man**

6.70. Tar-1.32

Paketet Tar innehåller ett arkiveringsprogram.

Förväntad byggtid: 2.0 SBU
Nödvändigt disk-utrymme: 45 MB

6.70.1. Installation av Tar

Förbered för kompilering:

```
FORCE_UNSAFE_CONFIGURE=1  \
./configure --prefix=/usr \
            --bindir=/bin
```

Vad alternativen betyder:

`FORCE_UNSAFE_CONFIGURE=1`
Tvingar testet för mknod att köras som root. Det betraktas allmänt som farligt att köra detta test som root, men då det körs på ett system som endast är delvis färdigbyggt är det ok att köra det på detta vis ändå.

Kompilera:

```
make
```

Kör test (3 SBU):

```
make check
```

Installera:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.32
```

6.70.2. Innehåll Tar

Installerade program: tar
Installerade mappar: /usr/share/doc/tar-1.32

Korta beskrivningar

tar Skapar, extraherar filer från, och listar innehållet i arkiv (även kallade tarballs)

6.71. Texinfo-6.7

Paketet Texinfo innehåller program för att läsa, skriva, och konvertera info sidor.

Förväntad byggtid: 0.7 SBU
Nödvändigt disk-utrymme: 116 MB

6.71.1. Installation av Texinfo

Förbered för kompilering:

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

Vad alternativen betyder:

`--disable-static`
I detta fall kommer huvudnivåns configure-skript att klaga på att detta är ett icke igenkänt alternativ, men configure-skriptet för XSParagraph känner igen det och använder det för att deaktivera installationen av en statisk XSParagraph.a till /usr /lib /texinfo.

Kompilera:

```
make
```

Kör test:

```
make check
```

Installera:

```
make install
```

Som extra tillval, installera komponenterna vilka tillhör en TeX installation:

```
make TEXMF=/usr/share/texmf install-tex
```

Vad alternativen betyder:

`TEXMF=/usr/share/texmf`
Makefile variabeln för TEXMF lagrar root-platsen för TeX-mappträdet, i det fall att exempelvis ett TeX paket installeras senare.

Info dokumentations systemet använder en textfil för att lagra meny entries. Filen återfinns i /usr /share /info /dir. Tyvärr och på grund av problem i makefilerna för diverse paket, kan detta system ibland hamna ur synk med info sidorna vilka är installerade på systemet. Ifall filen /usr /share /info /dir någonsin behöver rekonstrueras, kommer följande kommando att göra detta:

```
pushd /usr/share/info
rm -v dir
for f in *
  do install-info $f dir 2>/dev/null
done
popd
```

6.71.2. Innehåll Texinfo

Installerade program: info, install-info, makeinfo (länk till texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf, och texindex
Installerade bibliotek: MiscXS.so, Parsetexi.so, och XSParagraph.so (alla i /usr/lib/texinfo)
Installerade mappar: /usr/share/texinfo och /usr/lib/texinfo

Korta beskrivningar

info Används för att läsa info sidor, vilka ofta liknar man sidor men ofta går mycket djupare än att bara förklara vad alla de tillgängliga kommandorads alternativen är. Jämför t.ex **man bison** och **info bison**

install-info Används för att installera info sidor. Uppdaterar entries i index filen för **info**

makeinfo Översätter de givna Texinfo källkods dokumenten till info sidor, plain text, eller HTML

pdftexi2dvi Används för att formatera det givna Texinfo dokumentet till en PDF

pod2texi Omvandlar Pod till Texinfo format

texi2any Översätter Texinfo källkods dokumentation till diverse andra format

texi2dvi Används för att formatera det givna Texinfo dokumentet till en enhetsoberoende fil som kan skrivas ut

texi2pdf Används för att formatera det givna Texinfo dokumentet till en PDF

texindex Används för att sortera index filer för Texinfo

6.72. Vim-8.2.0190

Paketet Vim innehåller en kraftfull textredigerare.

Förväntad byggtid: 1.7 SBU
Nödvändigt disk-utrymme: 202 MB

Alternativ till Vim

Om du föredrar en annan redigerare - som Emacs, Joe, Nano - referera då till *http://www.linuxfromscratch.org/blfs/view/9.1/postlfs/editors.html* för instruktioner vad gäller installation.

6.72.1. Installation av Vim

Ändra först standard platsen för `vimrc` konfigurations filen till`/etc`:

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Förbered för kompilering:

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

Kompilera:

```
make
```

Som förberedelse för testen, säkerställ att användaren “nobody” kan skriva till trädet sources.

```
chown -Rv nobody .
```

Kör testen, som användare “nobody”:

```
su nobody -s /bin/bash -c "LANG=en_US.UTF-8 make -j1 test" &> vim-test.log
```

Testsviten skriver ut massor av binär data till skärmen. Detta kan skapa problem med den nuvarande terminalens inställningar. Problemet kan undvikas genom att omdirigera outputen till en logfil, som visas ovan. Ett framgångsrikt test kommer resultera i orden “ALL DONE” skrivna i logfilen.

Installera:

```
make install
```

Många användare är vana vid att använda **vi** istället för **vim**. För att tillåta exekvering av **vim** då användare av ren vana skriver **vi**, skapa en symlänk för både binärfilen samt man sidan:

```
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 installeras Vims dokumentation i /usr /share /vim. Följande symlänk tillåter att dokumentationen kan kommas åt via /usr /share /doc /vim-8.2.0190, vilket innebär att detta är konsistent med platsen för andra pakets dokumentation.

```
ln -sv ../vim/vim82/doc /usr/share/doc/vim-8.2.0190
```

Om ett X Window system ska installeras på LFS systemet, kan det vara nödvändigt att återkompilera Vim efter att ha installerat X. Med Vim inkluderas en GUI version av redigeraren, och denna version kräver att X samt ett antal övriga bibliotek finns installerade. För mer information vad gäller denna process, referera till Vims dokumentation sektionen för hur Vim installeras i boken BLFS. http://www.linuxfromscratch.org/blfs/view/9.1/postlfs/vim.html

6.72.2. Konfiguration av Vim

Som standard körs **vim** i vi-inkompatibelt tillstånd. Detta kan vara nytt för användare vilka tidigare har använt andra redigerare. Den “ickekompatibla” inställningen inkluderas nedan för att belysa faktumet att ett nytt beteende används. Det påminner även dem som vill ändra till “kompatibelt” tillstånd att det borde vara den första inställningen i config filen. Detta är nödvändigt då det ändrar andra inställningar, och overrides måste komma efter denna inställning. Skapa en standard konfigurationsfil för **vim** genom att köra följande kommando:

```
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
```

Inställningen *set nocompatible* får **vim** till att bete sig på ett mer användbart vis (standard) än i det vi-kompatibla tillståndet. Radera “no” för att behålla det gamla **vi** beteendet. “set backspace=2” inställningen tillåter backspacing över t.ex radbrytningar och autoindents. Parametern “syntax on” aktiverar vims syntax highlighting. “set mouse=” möjliggör korrekt inklistring av text med hjälp av musen, då man arbetar i chroot eller från en uppkopplad destination. Slutligen så innebär “if” tillstånden att vims gissning om vilken bakgrundsfärg som ska användas för en terminal, optimiseras.

Dokumentation för övriga alternativ kan erhållas med hjälp av:

```
vim -c ':options'
```

Kommentar

Som standard installerar Vim endast stavnings filer för Engelska. För att installera stavnings filer för ditt önskade språk, ladda ned *.spl och potentiellt även *.sug filerna för ditt språk och encoding, från: ftp://ftp.vim.org/pub/vim/runtime/spell/ och spara dem till /usr/share/vim/vim82/spell/ .

För att använda desssa stavningsfiler, konfigurera`/etc/vimrc`

```
set spelllang=en,ru
set spell
```

För mer information, referera till README filen som återfinns via URLen ovan.

6.72.3. Innehåll Vim

Installerade program: ex (länk till vim), rview (länk till vim), rvim (länk till vim), vi (länk till vim), view (länk till vim), vim, vimdiff (länk till vim), vimtutor, and xxd
Installerade mappar: /usr/share/vim

Korta beskrivningar

ex Startar **vim i** ex tillstånd

rview En begränsad version av **view**; inga skalkommandon kan startas och **view** kan inte låsas ned

rvim En begränsad version av **vim**; inga skalkommandon kan startas och **vim** kan inte låsas ned

vi Länk till **vim**

view Startar **vim** i läs-endast tillstånd

vim Redigeraren självt

vimdiff Redigerar två eller tre versioner av en fil med **vim** och visar skillnaderna

vimtutor Lär ut den grundläggande funktionaliteten inom **vim**

xxd Skapar en hex dump av den givna filen. Kan även göra det motsatta, och kan därför användas för binär patchning

6.73. Procps-ng-3.3.15

Paketet Procps-ng innehåller program för att monitorera processer

Förväntad byggtid: 0.1 SBU
Nödvändigt disk-utrymme: 17 MB

6.73.1. Installation av Procps-ng

Förbered för kompilering:

```
./configure --prefix=/usr                            \
            --exec-prefix=                           \
            --libdir=/usr/lib                        \
            --docdir=/usr/share/doc/procps-ng-3.3.15 \
            --disable-static                         \
            --disable-kill
```

Vad alternativen betyder:

`--disable-kill`
Deaktiverar byggandet av kommandot **kill,** vilket kommer installeras med paketet Util-linux.

Kompilera:

```
make
```

Testsviten behöver modifieras för LFS. Radera ett test som misslyckas när skripting inte använder en tty enhet, och fixa två andra. Kör testsviten med nedan kommandon:

```
sed -i -r 's|(pmap_initname)\\\$|\1|' testsuite/pmap.test/pmap.exp
sed -i '/set tty/d' testsuite/pkill.test/pkill.exp
rm testsuite/pgrep.test/pgrep.exp
make check
```

Installera:

```
make install
```

Flytta slutligen några nödvändiga bibliotek, så de kan nås även om `/usr` inte är monterad.

```
mv -v /usr/lib/libprocps.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libprocps.so) /usr/lib/libprocps.so
```

6.73.2. Innehåll Procps-ng

Installerade program: free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w, och watch
Installerade bibliotek: libprocps.so
Installerade mappar: /usr/include/proc och /usr/share/doc/procps-ng-3.3.15

Korta beskrivningar

free Rapporterar mängden av tillgängligt och använt minne (både fysiskt och swap) i systemet.

pgrep	Kolla upp processer baserat på deras namn och andra attribut
pidof	Rapporterar PIDs för de givna programmen
pkill	Skicka signaler till processer baserat på deras namn och andra attribut
pmap	Rapporterar minnes kartan för den givna processen
ps	Listar de nuvarande aktiva processerna
pwdx	Rapporterar den nuvarande arbetsmappen för en process
slabtop	Visar i realtid detaljerad kernel slab cache information
sysctl	Modifierar kernel parametrar vid run time
tload	Skriver en graf som visar det nuvarande ansträngnings genomsnittet för systemet
top	Visar en lista över de mest CPU intensiva processerna. Tillhandahåller en insyn i processoraktivitet i realtid.
uptime	Rapporterar hur länge systemet varit aktivt, hur många användare som är inloggade, och systemets resurs-användnings genomsnitt
vmstat	Rapporterar statistik för virtuellt minne, tillhandahållande information om processer, minne, sidor, block input/output, traps, samt CPU aktivitet.
w	Visar vilka användare som för tillfället är inloggade, var de är, och sedan när de är inloggade
watch	Kör ett givet kommando upprepade gånger, och skriver resultatet till skärmen så det går se hur output förändras.
`libprocps`	Innehåller funktionerna använda av de flesta program i detta paket

6.74. Util-linux-2.35.1

Paketet Util-linux innehåller diverse program. Bland dem återfinns verktyg för att hantera filsystem, konsoler, partitioner samt meddelanden.

Förväntad byggtid: 1.1 SBU
Nödvändigt disk-utrymme: 289 MB

6.74.1. FHS kommentarer

FHS rekommenderar användning av mappen `/var/lib/hwclock` istället för den traditionella `/etc` för placering av filen adjtime. Skapa först en mapp för att aktivera lagring för programmet **hwclock.**

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

6.74.2. Installation av Util-linux

Förbered för kompilering:

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime   \
            --docdir=/usr/share/doc/util-linux-2.35.1 \
            --disable-chfn-chsh  \
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
            --disable-runuser    \
            --disable-pylibmount \
            --disable-static     \
            --without-python     \
            --without-systemd    \
            --without-systemdsystemunitdir
```

Flaggorna --disable och --without förhindrar varningar angående byggandet av komponenter som kräver paket vilka inte återfinns i LFS eller som icke är konsistenta med program installerade av andra paket.

Kompilera:

```
make
```

Om önskat, kör testsviten som en ickeroot användare.

Varning

Att köra testsviten som root kan vara skadligt för ditt system. För att köra den, måste CONFIG_SCSI_DEBUG alternativet för kerneln vara tillgänglig i det nuvarande aktiva systemet, och måste byggas som en modul. Att bygga in den i kerneln kommer förhindra bootning. För en komplett körning av testsviten måste paket från BLFS installeras. Om önskat, kan testsviten köras efter att man rebootat in i det kompletta LFS systemet:

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

```
chown -Rv nobody .
su nobody -s /bin/bash -c "PATH=$PATH make -k check"
```

Installera:

```
make install
```

6.74.3. Innehåll Util-linux

Installerade program: addpart, agetty, blkdiscard, blkid, blkzone, blockdev, cal, cfdisk, chcpu, chmem, choom, chrt, col, colcrt, colrm, column, ctrlaltdel, delpart, dmesg, eject, fallocate, fdformat, fdisk, fincore, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipcs, isosize, kill, last, lastb (länk till last), ldattach, linux32, linux64, logger, look, losetup, lsblk, lscpu, lsipc, lslocks, lslogins, lsmem, lsns, mcookie, mesg, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, nsenter, partx, pivot_root, prlimit, raw, readprofile, rename, renice, resizepart, rev, rfkill, rtcwake, script, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff (länk till swapon), swapon, switch_root, taskset, ul, umount, uname26, unshare, utmpdump, uuidd, uuidgen, uuidparse, wall, wdctl, whereis, wipefs, x86_64, och zramctl

Installerade bibliotek: libblkid.so, libfdisk.so, libmount.so, libsmartcols.so, och libuuid.so

Installerade mappar: /usr/include/blkid, /usr/include/libfdisk, /usr/include/libmount, /usr/include/libsmartcols, /usr/include/uuid, /usr/share/doc/util-linux-2.35.1, och /var/lib/hwclock

Korta beskrivningar

addpart Informerar Linux kerneln om nya partitioner

agetty Öppnar en tty port, frågar efter ett login namn, och kör sedan programmet **login**

blkdiscard Raderar sektorer på en enhet

blkid Ett kommandorads verktyg för att lokalisera och skriva block enhets attribut

blkzone Kör zone kommando på den givna block enheten

blockdev Tillåter användare att kalla block enheten ioctls via kommandoraden

cal Visar en simpel kalender

cfdisk Manipulerar partitions table för den givna enheten

chcpu Modifierar CPUns tillstånd

chmem Konfigurerar minne

choom Visar och justerar OOM-killer resultat

chrt Manipulerar realtids attribut för en process

col Filtrerar ut omvända rad feeds

colcrt Filtrerar **nroff** output för terminaler som saknar viss funktionalitet, som t.ex overstriking / half-lines

colrm Filtrerar ut de givna kolumnerna

column Formaterar en given fil till multipla kolumner

ctrlaltdel Definierar Ctrl+Alt+Del kombinationen till en hård eller mjuk reset

delpart Frågar Linux kernel ifall den kan radera en partition

dmesg Dumpar kernelns boot meddelanden

eject Öppnar CDROM/DVD eller annan uttagbar media

fallocate Förallokerar utrymme till en fil

fdformat Lågnivå formaterar partitions table för den givna enheten

fdisk Manipulerar partitions table för den givna enheten

fincore Räknar antalet sidor med fil innehåll i core

findfs Hittar ett filsystem med hjälp av label eller Universally Unique Identifier (UUID)

findmnt Kommandorads gränssnitt för biblioteket libmount så det går arbeta med mountinfo, fstab, mtab filer

flock Erhåller ett fil lås, och exekverar sedan ett kommando med låset kvar

fsck Används för att dubbelkolla, och alternativt reparera filsystem

fsck.cramfs Utför en consistency check mot Cramfs filsystemet på den givna enheten

fsck.minix Utför en consistency check mot Minix filsystemet på den givna enheten

fsfreeze En väldigt simpel wrapper omkring FIFREEZE/FITHAW ioctl kernelns drivrutins operationer

fstrim Slänger oanvända block på ett monterat filsystem

getopt Söker igenom alternativ i den givna kommandoraden

hexdump Dumpar den givna filen i hexadecimal eller annat specifierat format

hwclock Läser eller definierar systemets hårdvaru klocka Real-Time Clock (RTC) eller Basic Input-Output System(BIOS) clock

i386 Symbolisk länk till setarch

ionice Gets eller sets io scheduling klass samt ett programs prioritet

ipcmk Skapar diverse IPC resurser

ipcrm Raderar den givna Inter-Process Communication (IPC) resursen

ipcs Tillhandahåller IPC status information

isosize Rapporterar storleken av dett iso9660 filsystem

kill Skickar signaler till processer

last Visar vilka användare som senast loggade in och ut, genom att sök filen`/var/log/wtmp` ; visar även system uppstarter, nedstängningar, och run-time förändringar

lastb Visar misslyckade inloggningförsök, loggade i`/var/log/btmp`

ldattach Fäster en rad disciplin till en seriell rad

linux32 En symbolisk länk till setarch

linux64 En symbolisk länk till setarch

logger Skriver det givna meddelandet till systemlogen

look Visar rader som börjar med den givna strängen

losetup Konfigurerar och kontrollerar loop enheter

lsblk Listar information om valda block enheter i ett trädliknande format

lscpu Skriver arkitektur information om CPUn

lsipc Skriver information om IPC funktioner för tillfället implementerade i systemet

lslocks Listar lokala system lås

lslogins	Listar information of användare, grupper, och systemkonton
lsmem	Listar spektrumen av tillgängligt minne, samt deras uppkopplade status
lsns	Listar namespaces
mcookie	Genererar magic cookies (128-bit slumpmässiga hexadecimala nummber) för **xauth**
mesg	Kontrollerar ifall andra användare kan skicka meddelanden till den nuvarande användarens terminal
mkfs	Bygger ett filsystem på en enhet (vanligtvis en hårddisks partition)
mkfs.bfs	Skapar ett Santa Cruz Operations (SCO) bfs filsystem
mkfs.cramfs	Skapar ett cramfs filsystem
mkfs.minix	Skapar ett Minix filsystem
mkswap	Initialiserar den givna enheten eller filen till att användas som swap område
more	Ett filter för att scrolla igenom text en skärm i taget
mount	Bifogar filsystemet på en given enhete till en specifierad mapp i filsystems trädet.
mountpoint	Dubbelkollar ifall mappen är en monteringspunkt
namei	Visar de symboliska länkarna i de givna sökvägsnamnen
nsenter	Kör ett program med namespaces för andra processer
partx	Informerar kerneln om närvaron samt numreringen av on-disk partitioner
pivot_root	Gör det givna filsystemet till det nya root filsystemet för den nuvarande processen
prlimit	Get och set resursbegränsningar för en process
raw	Bind en Linux raw character enhet till en block enhet
readprofile	Läser kernel profilierings information
rename	Byter namn på de givna filerna, ersättande en given sträng med en annan
renice	Ändrar prioriteten för aktiva processer
resizepart	Ber Linux kerneln att ändra storlek på en partition
rev	Reverses raderna i en given fil.
rkfill	Verktyg för att aktivera och deaktivera trådlösa enheter
rtcwake	Används för att låte systemet gå in i Sleep stadiet, tills en specifierad tid för det att vakna upp
script	Skapar ett typescript av en terminals session
scriptreplay	Spelar upp typescripts med hjälp av tidsbaserad information
setarch	Ändrar rapporterad arkitektur i en ny program miljö samt definierar personlighets flaggor
setsid	Kör det givna programmet i en ny session
setterm	Sets terminalattribut
sfdisk	En disk partitions table manipulator
sulogin	Tillåter `root` att logga in; kallas vanligtvis av **init** då systemet äntrar single user tillstånd
swaplabel	Tillåter att man ändrar swaparea UUID och label
swapoff	Deaktiverar enheter och filer för paging och swapping
swapon	Aktiverar enheter och filer för paging och swapping samt listar enheter och filer som för närvarande används

switch_root	Switches to another filesystem as the root of the mount tree
tailf	Spårar hur en logfil växer. Visar de 10 sista raderna i en logfil, och visar sedan nya entries allteftersom de skrivs till logfilen.
taskset	Erhåller eller definierar en CPU affinity för en proccess
ul	Ett filter för att översätta underscores till escape sekvenser vilka indikerar underlining för terminalen som används.
umount	Kopplar bort ett filsystem från systemets filträd
uname26	Symbolisk länk till setarch
unshare	Kör ett program med vissa namespaces ej delade av föräldern
utmpdump	Visar innehålet i en given login fil, men i ett mer användarvänligt format
uuidd	En daemon som används av biblioteket UUID för att generera tisdbaserade UUIDs på ett säkert och unik-garanterat vis.
uuidgen	Skapar nya UUIDs. Varje UUID kan rimligen betraktas som unikt ibland alla UUIDs vilka har skapats, på både det lokala systemet samt på externa system, i det förflutna samt framtiden.
uuidparse	Ett verktyg för att parsea unika identifikationer
wall	Visar innehållet i en fil eller, som standard, dess standard input, på terminalerna av alla för tillfället inloggade användare.
wdctl	Visar hårdvaru watchdog status
whereis	Rapporterar binärens plats, källa, och man page för det givna kommandot
wipefs	Raderar en filsystems-signatur från en enhet
x86_64	Symbolisk länk till setarch
zramctl	Program för att konfigurera och kontrollera zram (kompimerad ram disk) enheter
`libblkid`	Innehåller rutiner för enhetsidentifikation samt token extrahering
`libfdisk`	Innehåller rutiner för att manipulera partition tables
`libmount`	Innehåller rutiner för block enhets montering och avmontering
`libsmartcols`	Innehåller rutiner för att underlätta skärmoutput i tabulär form
`libuuid`	Innehåller rutiner för att generera unika identifierare till objekt vilka ska vara tillgängliga även bortom det lokala systemet

6.75. E2fsprogs-1.45.5

Paketet E2fsprogs innehåller verktyg för att hantera filsystemet `ext2`. Det stödjer även filsystemen `ext3` och `ext4` journaling file systems.

Förväntad byggtid: 1.6 SBU
Nödvändigt disk-utrymme: 108 MB

6.75.1. Installation av E2fsprogs

Dokumentationen för E2fsprogs rekommenderar att paketet byggs i en undermapp i source trädet:

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

Förbered för kompilering:

```
../configure --prefix=/usr           \
             --bindir=/bin           \
             --with-root-prefix=""   \
             --enable-elf-shlibs     \
             --disable-libblkid      \
             --disable-libuuid       \
             --disable-uuidd         \
             --disable-fsck
```

Vad alternativen betyder:

`--with-root-prefix=""` and *`--bindir=/bin`*
Vissa program (som **e2fsck**) betraktas som nödvändiga program. Då, exempelvis, /usr inte är monterad, behöver dessa program ändå finnas tillgängliga. Det hör hemma i mappar som /lib och /sbin. Ifall detta alternativ inte skickas till E2fsprogs configure, installeras dessa program i /usr.

`--enable-elf-shlibs`
Skapar delade bibliotek vilka vissa av programmen i detta paket användare.

`--disable-*`
Förhindrar E2fsprogs från att bygga och installera biblioteken libuuid, libblkid, daemonen uuid, och wrappern **fsck**. Util-Linux installerar mer uppdaterade versioner.

Kompilera:

```
make
```

Kör test:

```
make check
```

Ett av testen för E2fsprogs kommer försöka allokera 256 MB minne. Om du behöver använda en swap disk för detta, refera till sektion 2.5 samt 2.7.

Installera:

```
make install
```

Gör de installerade statiska biblioteken skrivbara så att felsöknings symboler senare kan tas bort:

```
chmod -v u+w /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Detta paket installerar en gzippad .info fil men uppdatera inte den systemvida dir filen. Extrahera denna fil och uppdatera sedan systemets dir-fil med hjälp av nedan kommando:

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir /usr/share/info/libext2fs.info
```

Om önskat, skapa och installera extra dokumentation:

```
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
```

6.75.2. Innehåll E2fsprogs

Installerade program:	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, och tune2fs
Installerade bibliotek:	libcom_err.so, libe2p.so, libext2fs.so, och libss.so
Installerade mappar:	/usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/ss, /usr/lib/e2fsprogs, /usr/share/et, och /usr/share/ss

Korta beskrivningar

badblocks	Söker igenom en enhet (vanligtvis en disk partition) efter dåliga block
chattr	Ändrar attributen för filer på ett ext2 filsystem; ändrar även ext3 filsystem, som är den journalbetonade versionen av ext2.
compile_et	En error table kompilator; omvandlar ett table av error-code namn och meddelanden till en C source fil lämplig för användning med com_err biblioteket.
debugfs	En filsystems felsökare. Kan användas för att undersöka och ändra tillstånd för ett ext2 filsystem
dumpe2fs	Prints super block and blocks group information for the file system present on a given device
e2freefrag	Reports free space fragmentation information
e2fsck	Används för att dubbelkolla, samt vid behov reparera ext2 och ext3 filsystem
e2image	Används för att spara kritisk ext2 filsystems data till en fil
e2label	Visar eller ändrar filsystems etiketten för ext2 filsystemet tillgängligt på en given enhet
e2mmpstatus	Dubbelkollar MMP status för ett ext4 filsystem
e2scrub	Dubbelkollar innehållet i ett monterat ext[234] filsystem
e2scrub_all	Dubbelkollar alla monterade ext[234] filsystem efter errors
e2undo	Spelar upp loggen undo_log för ett ext2/ext3/ext4 filsystem på en enhete [Detta kan användas för att göra en e2fsprogs misslyckad operation ogjord].
e4crypt	Ext4 filsystem krypterings verktyg
e4defrag	Online defragmenterare för ext4 filsystem

filefrag	Rapporterar hur illa fragmenterad en fil är
fsck.ext2	Kollar som standard `ext2` filsystem och är en hård länk till **e2fsck**
fsck.ext3	Kollar som standard `ext3` filsystem och är en hård länk till **e2fsck**
fsck.ext4	Kollar som standard `ext4` filsystem och är en hård länk till **e2fsck**
logsave	Sparar ett kommandos output till en logfil
lsattr	Listar attributen för filer på ett sekundärt förlängt filsystem
mk_cmds	Omvandlar ett table av kommandonamn och hjälpmeddelanden till en C sourcefil lämplig för användning med libss undersystem bibliotek
mke2fs	Skapar ett `ext2` / `ext3` filsystem på den givna enheten
mkfs.ext2	Skapar som standard `ext2` filsystem och är en hård länk till **mke2fs**
mkfs.ext3	Skapar som standard `ext3` filsystem och är en hård länk till **mke2fs**
mkfs.ext4	Skapar som standard `ext4` filsystem och är en hård länk till **mke2fs**
mklost+found	Skapar mappen “lost+found” på ett ext2 filsystem; förallokerar disk block till denna mapp för att underlätta för **e2fsck**.
resize2fs	Kan användas för att krympa eller expandera ett `ext2` filsystem
tune2fs	Justerar påverkbara filsystems parametrar i ett `ext2` filsystem
`libcom_err`	Den vanliga error visnings rutinen
`libe2p`	Används av **dumpe2fs**, **chattr**, och **lsattr**
`libext2fs`	Innehåller rutiner för att möjliggöra användar-nivås program att manipulera ett ext2 filsystem
`libss`	Används av **debugfs**

6.76. Sysklogd-1.5.1

Paketet Sysklogd innehåller program för att logga systemmeddelanden, som de kerneln skickar då något oväntat sker.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 0.6 MB

6.76.1. Installation av Sysklogd

Fixa först några problem som leder till segmentations fel under vissa tillstånd i klogd och fixa även en utdaterad program construct:

```
sed -i '/Error loading kernel symbols/{n;n;d}' ksym_mod.c
sed -i 's/union wait/int/' syslogd.c
```

Kompilera:

```
make
```

Detta paket inkluderar inte en testsvit.

Installera:

```
make BINDIR=/sbin install
```

6.76.2. Konfigurera Sysklogd

Skapa en ny `/etc/syslog.conf` fil genom att köra följande:

```
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
```

6.76.3. Innehåll Sysklogd

Installerade program: klogd och syslogd

Korta beskrivningar

klogd En system daemon för att fånga upp och logga kernelmeddelanden

syslogd Loggar meddelanden vilka systemets program erbjuder för att loggas. Varje loggat meddelande innehåller som minimum en datumsetikett och ett värdnamn, och normalt sett även programmets namn men detta beror på hur stor tillit daemon är instruerad att använda sig av.

6.77. Sysvinit-2.96

Paketet Sysvinit innehåller program som kontrollerar uppstart, körning, samt nedstängning av systemet.

Förväntad byggtid: Mindre än 0.1 SBU
Nödvändigt disk-utrymme: 1.4 MB

6.77.1. Installation av Sysvinit

Applicera först en patch som tar bort ett antal program installerade av andra paket, som tydliggör ett meddelandes betydelse, samt fixar en kompilatorvarning:

```
patch -Np1 -i ../sysvinit-2.96-consolidated-1.patch
```

Kompilera:

```
make
```

Detta paket inkluderar inte en testsvit.

Installera:

```
make install
```

6.77.2. Innehåll Sysvinit

Installerade program: bootlogd, fstab-decode, halt, init, killall5, poweroff (länk till halt), reboot (länk till halt), runlevel, shutdown, och telinit (länk till init)

Korta beskrivningar

bootlogd Loggar uppstarts meddelanden till en logfil

fstab-decode Kör ett kommando med fstab-encoded argument

halt Åberopar vanligtvis **shutdown** med alternativet -h, förutom då redan i run-level 0, och specifierar sedan för kerneln att få systemet till halt; noterar i filen /var /log /wtmp att systemet tas ned

init Denna första process som startas då kerneln har initialiserat hårdvaran. Tar över uppstarts processen och startar alla processer vilka specifieras i dess konfigurationsfil.

killall5 Skickar en signal till alla processer, bortsett från processerna i dess egen session så att det inte dödar sitt föräldraskal.

poweroff Specifierar för kerneln att få systemet till halt och stänga av datorna (referens **halt**)

reboot Specifierar för kerneln att starta om systemet (referens **halt**)

runlevel Rapporterar tidigare och nuvarande run-level, som noterat i senaste run-level loggningen i `/var/run/utmp`

shutdown Stänger ned systemet på ett säkert sätt, genom att signalera alla processer och användare om detta

telinit Specifierar för **init** vilken run-level att ändra till

6.78. Eudev-3.2.9

Paketet Eudev innehåller program för dynamiskt skapande av enhets noder.

Förväntad byggtid: 0.2 SBU
Nödvändigt disk-utrymme: 83 MB

6.78.1. Installation av Eudev

Förbered för kompilering:

```
./configure --prefix=/usr           \
            --bindir=/sbin          \
            --sbindir=/sbin         \
            --libdir=/usr/lib       \
            --sysconfdir=/etc       \
            --libexecdir=/lib       \
            --with-rootprefix=      \
            --with-rootlibdir=/lib  \
            --enable-manpages       \
            --disable-static
```

Kompilera:

```
make
```

Skapa ett par mappar som behövs för att utföra testen korrekt, men som även kommer användas i själva installationen:

```
mkdir -pv /lib/udev/rules.d
mkdir -pv /etc/udev/rules.d
```

Kör test:

```
make check
```

Installera:

```
make install
```

Installera ett antal standardregler och understödsfiler vilka är användbara i en LFS miljö:

```
tar -xvf ../udev-lfs-20171102.tar.xz
make -f udev-lfs-20171102/Makefile.lfs install
```

6.78.2. Konfigurera Eudev

Information om hårdvaru enheter återfinns i mapparna /etc /udev /hwdb.b och /lib /udev /hwdb.d. Denna information måste kompileras till en binär databas /etc /udev /hwdb.bin. Skapa den första instansen av databasen:

```
udevadm hwdb --update
```

Detta kommando måste köras varje gång som hårdvaru informationen uppdateras.

6.78.3. Innehåll Eudev

Installerade program: udevadm och udevd
Installerade bibliotek: libudev.so
Installerade mappar: /etc/udev, /lib/udev, och /usr/share/doc/udev-udev-lfs-20171102

Korta beskrivningar

udevadm Generiskt udev administrations verktyg. Kontrollerar udev daemonen, tillhandahåller info från udev databasen, monitorerar händelser, väntar på uevents att slutföras, testar udev konfiguration, och triggar uevents på en given enhet.

udevd En daemon som listar uevents på netlink socketen, skapar enheter och kör konfigurerade externa program i respons till dessa uevents.

`libudev` Ett biblioteks gränssnitt för udev enhets information

`/etc/udev` Innehåller Udev konfigurations filer, enhets tillstånd, samt regler för namngivning av enheter

6.79. Angående felsöknings-symboler

De flesta program och bibliotek kompileras som standard med felsökningssymboler inkluderade (**gcc** -g alternativ). Detta innebär att då ett program eller bibliotek vilket kompilerades med sådana symboler felsöks, kan felsökaren tillhandahålla information om namn på rutiner och variabler, och inte bara minnes adresser.

Inkluderingen av dessa symboler leder dock till att ett program eller bibliotek blir betydligt större. Nedan är exempel på hur stor skillnad det kan bli.

- En **bash** binär med felsöknings symboler: 1200 KB
- En **bash** binär utan felsöknings symboler: 480 KB
- Glibc och GCC filer (`/lib` och `/usr/lib`) med felsöknings symboler: 87 MB
- Glibc och GCC filer utan felsöknings symboler: 16 MB

Slutgiltig storlek kan variera beroende på vilken kompilator samt vilket C bibliotek som användes, men då man jämför storleken för program med eller utan felsöknings symboler, kommer skillnad vanligtvis vara en faktor mellan 2 och 5.

Eftersom de flesta användare aldrig kommer att använda en felsökare på sitt systems mjukvara, kan mycket utrymme sparas genom att radera dessa symboler. Nedan visar hur man tar bort felsöknings symboler i program och bibliotek.

6.80. Minimera igen

Denna sektion är ett fritt tillval. Om den tänkta användaren inte är en programmerare och inte planerar att köra någon felsökning på systemets mjukvara, kan systemets storlek minskas med ca 90mb genom att ta bort alla felsöknings symboler. Detta leder inte till något annat än att det inte går att felsöka systemets mjukvara lika ingående längre.

De flesta personer vilka använder kommandona nedan upplever inte några problem. Det är däremot enkelt att få med en felstavning vilken leder till att hela systemet görs oanvändbart. Det är därför en bra idé att säkerhetskopiera det nuvarande systemet innan det att du kör **strip** kommandona.

Placera först felsöknings symbolerna för valda bibliotek i separata filer. Denna felsöknings information behövs om man i BLFS ska köra regressions test vilka använder *valgrind* eller *gdb*.

```
save_lib="ld-2.31.so libc-2.31.so libpthread-2.31.so libthread_db-1.0.so"

cd /lib

for LIB in $save_lib; do
    objcopy --only-keep-debug $LIB $LIB.dbg
    strip --strip-unneeded $LIB
    objcopy --add-gnu-debuglink=$LIB.dbg $LIB
done

save_usrlib="libquadmath.so.0.0.0 libstdc++.so.6.0.27
             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
    strip --strip-unneeded $LIB
    objcopy --add-gnu-debuglink=$LIB.dbg $LIB
done

unset LIB save_lib save_usrlib
```

Innan det att stripping genomförs, säkerställ att inga av de binärer vilka ska rensas från felsöknings symboler för tillfället körs:

```
exec /tools/bin/bash
```

Det går nu på ett säkert vis genomföra stripping på bibliotek och binärer:

```
/tools/bin/find /usr/lib -type f -name \*.a \
   -exec /tools/bin/strip --strip-debug {} ';'

/tools/bin/find /lib /usr/lib -type f \( -name \*.so* -a ! -name \*dbg \) \
   -exec /tools/bin/strip --strip-unneeded {} ';'

/tools/bin/find /{bin,sbin} /usr/{bin,sbin,libexec} -type f \
    -exec /tools/bin/strip --strip-all {} ';'
```

Det kommer rapporteras att ett stort antal filers format inte kan igenkännas. Dessa varningar kan ignoreras. Dessa varningar indikerar att de nämnda filerna är skript och inte binärer.

6.81. Städa upp

Städa sedan upp diverse filer som uppstod till följd av testen:

```
rm -rf /tmp/*
```

Logga nu ut och återvänd till chroot miljön med ett uppdaterat chroot kommando. Använd framöver detta uppdaterade chroot kommando varje gång du behöver återvända till chroot miljön efter att ha lämnat den:

```
logout

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

Anledning till detta är att verktygen i /tools inte längre är nödvändiga. Du kan nu därför radera mappen /tools om du vill.

Kommentar

Att radera /tools kommer även ta bort de temporära kopiorna av Tcl, Expect och DejaGNU vilka användes för att köra toolchain testen. Om du behöver dessa program framöver, behöver de återkompileras och även installeras. Boken BLFS innehåller instruktioner för detta (http://www.linuxfromscratch.org/blfs/).

Ifall de virtuella kernel filsystemen kanske har avmonterats, vare sig manuellt eller via en omstart, säkerställ att dessa filsystem då du återvänder till croot är monterade. Denna process förklarades i sektion 6.2.2 samt sektion 6.2.3.

Det finns flertalet statiska bibliotek vilka krävdes för att understödja regressions testen för flertalet tidigare paket. Dessa bibliotek installerades med binutils, bzip2, e2fsprogs, flex, libtool, och zlib. Radera dem om du vill:

```
rm -f /usr/lib/lib{bfd,opcodes}.a
rm -f /usr/lib/libbz2.a
rm -f /usr/lib/lib{com_err,e2p,ext2fs,ss}.a
rm -f /usr/lib/libltdl.a
rm -f /usr/lib/libfl.a
rm -f /usr/lib/libz.a
```

Det finns även flertalet filer vilka är installerade i /usr /lib och /usr /libexec mapparna med en filnamns förlängning av .la. Dessa är “libtool arkiv” filer och behövs generellt sätt inte på ett Linux system. Inga av dessa är av behov längre. Radera dem med nedan kommando:

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

För mer information om libtool arkivfiler, se *BLFS sektionen "About Libtool Archive (.la) files"*.

Chapter 7. Systemkonfiguration

7.1. Introduktion

Att starta upp ett Linux system inkluderar ett antal uppgifter. Processen måste montera både virtuella samt verkliga filsystem, initialisera enheter, aktiver swap, dubbelkolla filsystemens integritet, montera swap partitioner och filer, definiera systemklockan, starta nätverk, starta daemoner, och slutföra diverse andra uppgifter.
För att denna process ska vara framgångsrik måste den vara organiserad så att allt utförs i den korrekta ordningen, samtidigt som uppgifterna utförs så snabbt som möjligt.

7.1.1. System V

System V är den klassika uppstarts-process som använts i Unix och Unix-liknande system såsom Linux, sedan ca 1983. Det består av ett litet program, **init**, vilket startar upp program såsom **login** (via getty) och kör ett skript. Detta skript, vanligtvis benämnt **rc**, kontrollerar utförandet av en samling skript vilka utför uppgifterna som initialiserar systemet.

Programmet **init** kontrolleras av filen`/etc/inittab` ich är organiserat i run levels som kan köras av användaren:

```
0 — halt
1 — Single user mode
2 — Multiuser, without networking
3 — Full multiuser mode
4 — User definable
5 — Full multiuser mode with display manager
6 — reboot
```

Den vanliga standard run level är 3 eller 5.

Fördelar

- Etablerat system som det finns mycket information om vad gäller hur det fungerar.
- Enkelt att anpassa efter egna önskemål.

Nackdelar

- Långsammare att starta upp. Ett medium hastighets LFS system tar 8-12 sekunder, när uppstartstid mäts från det första kernelmeddelandet till login prompten. Nätverks funktionalitet etableras vanligtvis ca 2 sekunder efter login prompt.
- Seriellt processerande av uppstarts uppgifter. Relaterat till den förra punkten. En fördröjning i individuella processer, såsom en filsystems kontroll, saktar upp hela uppstarts processen.
- Stödjer inte avancerad funktionalitet likt kontrollgrupper (cgroups), och per-user fair share scheduling.
- Att lägga till skript kräver manuella beslut vad gäller dessa statiska sekvenser.

7.2. LFS-Bootscripts-20191031

Paketet LFS-Bootscripts en samling skript som startar/stoppar LFS systemet vid uppstart/nedstängning. Konfigurationsfilerna och procedurerna som krävs för att anpassa uppstartsprocessen, beskrivs i de följande sektionerna.

Förväntad byggtid: mindre än 0.1 SBU
Nödvändigt disk-utrymme: 244 KB

7.2.1. Installation av LFS-Bootscripts

Installera:

```
make install
```

7.2.2. Innehåll LFS-Bootscripts

Installerade skript: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountvirtfs, network, rc, reboot, sendsignals, setclock, ipv4-static, swap, sysctl, sysklogd, template, udev, och udev_retry
Installerade mappar: /etc/rc.d, /etc/init.d (symbolisk länk), /etc/sysconfig, /lib/services, /lib/lsb (symbolisk länk)

Korta beskrivningar

checkfs Dubbelkollar filsystemens integritet innan att de monteras (med undantag för journal samt nätverksbaserade filsystem).

cleanfs Raderar filer som inte borde sparas emellan omstarter, såsom de i /var /run och /var /lock. Återskapar /var /run /utmp och raderar de potentiellt närvarande filerna /etc /nologin, /fastboot, och /forcefsck.

console Laddar det korrekta keymap table för den önskade tangentbords layouten. Specifierar även skärm-font

functions Innehåller vanliga funktioner, som error och status kollar, som används av flera bootscripts

halt För systemet till halt

ifdown Stoppar en nätverksenhet

ifup Initialiserar en nätverksenhet

localnet Startar upp systemets värdnamn samt lokala loopback enhet

modules Laddar kernelmoduler listade i`/etc/sysconfig/modules`, genom att använda argument vilka även finns återgivna där

mountfs Monterar alla filsystem, bortsett från de som är märkta *noauto* eller är nätverksbaserade

mountvirtfs Monterar virtuella kernel filsystem, som t.ex `proc`

network Startar upp nätverks gränssnitt, såsom nätverkskort, och startar upp en potentiell standard gateway

rc Master run-level kontroll skriptet; det är ansvarigt för att köra alla andra bootskript ett efter ett, i en sekvens som bestäms av namnen på de symboliska länkar som processas

reboot Startar om systemet

sendsignals Säkerställer att varje process är avslutad innan systemet startar om eller kommer till halt

setclock Återställer kernelklockan till lokal tid i det fall att klockan inte är satt till UTC

ipv4-static Tillhandahåller funktionaliteten som krävs för att bestämmar IP adressen för ett nätverks gränssnitt.

swap Aktiverar och deaktiverar swap filer och partitioner

sysctl Laddar systemkonfigurationsvärden från `/etc/sysctl.conf` , ifall den filen existerar, till den aktiva kerneln.

sysklogd Startar och stoppar log daemonerna för systemet och kerneln

template En mall för att skapa anpassade bootskript för andra daemoner

udev Förberedr `/dev` och startar udev.

udev_retry Testar igen att köra misslyckade udev uevents, och kopiererar genererade regelfiler från /run /udev till /etc /udev /rules.d om så behövs.

7.3. Översikt gällande enhets och modul-hantering

i kapitel 6 installerade vi paketet Udev då eudev byggdes. Innan vi går in i detalj på hur hur detta paket funkar, är det tid för en kort sammanfattning vad gäller tidigare metoder för att hantera enheter.

Linuxsystem använde traditionellt sett en statisk metod för att skapa enheter, med vilken ett stort antal enhetsnoder skapades under /dev (ibland tusentals noder), oavsett ifall de motsvarande hårdvaruenheterna faktiskt existerade. Detta gjordes vanligtvis via ett MAKEDEV skript, vilket innehåller ett antal anrop till programmet **mknod** som inkluderar de relevanta enhetsnumren för varje enhet som kan tänkas existera.

Genom att använda metoden Udev, skapas endast noder för de hårdvaruenheter som upptäcks av kerneln. Eftersom dessa noder kommer skapas varje gång som systemet bootar, kommer de sparas på ett devtmpfs filsystem (ett virtuellt filsystem som existerar helt och hållet i systemminnet). Enhetsnoder kräver inte mycket minne.

7.3.1. Historia

I februari 2000 sammanfogades ett filsystem vid namn devfs med 2.3.46 kerneln, och gjordes tillgänligt via 2.4 versionen av stabila kernels. Även ifall metoden fanns tillgänglig i själva kerneln, erhöll denna metod att dynamiskt skapa enhetsnoder aldrig något omfattande stöd från de mest inflytelserika kernel utvecklarna.
Det huvudsakliga problemet med denna variant var hur den hanterade enhets identifiering, skapande, och namngivning. Den senare orsaken, enhetsnods namngivning, var antagligen den mest kritiska. Det är generellt sett accepterat att ifall enhetsnamn tillåts att konfigureras, då bör enhets-namngivnings policyn stå under sysadmins kontroll, och inte påtvingad systemet av utvecklarna. Devfs filsystem led också av race conditions som var del av själva systemdesignen och kunde inte fixas utan omfattande förändringar av kerneln. Devfs markerades som utfasat – pga inget underhåll – och togs bort från kerneln i Juni 2006.
Med utvecklingen av det ostabila 2.5 kernel trädet, senare släppt som 2.6 serien av stabila kernels, skapades ett nytt virtuellt filsystem kallat sysfs. Sysfs jobb är att exportera en översyn av systemets hårdvaru-konfiguration till userspace processer. Med denna userspace synliga representationen, blev möjligheten att utveckla en userspace ersättning för devfs men realistiskt förankrad.

7.3.2. Udev Implementation

7.3.2.1. Sysfs

Filsystemet sysfs nämndes ovan, och man kan undra hur sysfs kan avgöra vilka enheter som finns tillgängliga på ett system samt vilka enhetsnummer som borde användas för dessa. Drivrutiner som har kompilerats direkt in i kerneln registerar sina objekt med ett sysfs (devtmpfs internt) allteftersom de upptäcks av kerneln. För drivrutiner kompilerade som moduler, kommer denna registrering ske då modulen är laddad. När sysfs filsystemet väl är laddat (på /sys), kommer data vilken drivrutinerna registrerar med sysfs att bli tillgänglig för userspace processer och udev att processas (inkluderat modifikationer gjorda på enhetsnoder).

7.3.2.2. Nod-skapande för enhet

Enhetsfiler skapas av kerneln med hjälp av filsystemet devtmpfs. Drivrutiner som önskar registrera en enhetsnod går via devtmpfs (via drivrutins kärnan) för att göra detta. När en devtmpfs instans monteras på /dev, kommer enhetsnoden att initialt skapas med ett bestämt namn, rättigheter, samt ägare.

En kort stund därefter kommer kerneln att skicka en uevent till **udevd**. Baserat på reglerna specifierade i filerna i /etc /udev /rules.d, /lib /udev /rules.d, och /run /udev /rules.d mapparna, kommer **udevd** att skapa ytterligare symlänkar till enhetsnoden, eller ändra dess rättigheter, ägare, grupp, eller modifiera den interna **udevd** databas entryn (namnet) för det objektet.

Reglerna i dessa tre mappar är numrerade och alla tre mappar är sammanfogade med varandra. Ifall **udevd** inte kan hitta en regel för enheten vilken skapas, kommer rättigheter och ägarskap lämnas kvar som vad devtmpfs skapade den med.

7.3.2.3. Laddning av moduler

Enhetsdrivrutiner kompilerade som moduler kan ha alias inbyggda. Aliasen är synliga i output från **modinfo** programmet och är vanligtvis relaterade till bus-specifika identifierare för enheter stödda av en modul. Exempelvis snd-fm801 drivrutinen stödjer PCI enheter med vendor ID 0x1319 och enhets ID 0x0801, och har ett alias “pci:v00001319d00000801sv*sd*bc04sc01i*”. För de flesta enheter, exporterar bus-drivrutinen aliaset för den drivrutin vilken kommer hantera enheten via sysfs. T.ex filen /sys/bus/pci/devices/0000:00:0d.0/modalias kanske kommer innehålla strängen “pci:v00001319d00000801sv00001319sd00001319bc04sc01i00”. Standard reglerna tillhandahållna med Udev kommer få **udevd** att åberopa **/sbin /modprobe** med innehållet i MODALIAS uevent miljövariabeln (vilket borde vara samma innehåll som i MODALIAS filen i sysfs), därmed laddande alla moduler vars alias matchar denna sträng efter wildcard expansion.

För detta exempel innebär det att, utöver *snd-fm801*, den utfasade (och oönskade) *forte* drivrutinen kommer laddas ifall den är tillgänglig. Se nedan hur man hindrar att oönskade drivrutiner laddas.

Kerneln själv är också kapabel att ladda moduler för nätverks protokoll, filsystem och NLS support.

7.3.2.4. Att hantera Hotpluggable/Dynamic enheter

När du pluggar in en enhet, såsom en USB mp3-spelare, kommer kerneln reagera på att enheten kopplas in och därför generera ett uevent. Detta uevent hanteras sedan av **udevd** på det sätt vilket beskrivs ovan.

7.3.3. Problem med att ladda moduler och skapa enheter

Det finns ett antal problem som kan uppstå när det kommer till att automatiskt skapa enhetsnoder.

7.3.3.1. En kernel-modul laddas inte automatiskt

Udev kommer endast ladda en modul om den har ett bus-specifikt alias och bus-drivrutinen korrekt exporterar nödvändiga alias till sysfs. i andra fall bör man arrangera laddandet av moduler på andra sätt. Med Linux-5.5.3 är det känt att Udev laddar korrekt skrivna drivrutiner för INPUT, IDE, PCI, USB, SCSI, SERIO, och FireWire enheter.

För att avgöra ifall den drivrutin du behöver har stöd för Udev, kör **modinfo** med modulnamnet som argument. Försök nu att lokalisera enhetsmappen under /sys /bus och dubbelkolla ifall det finns en modalias fil där.

Ifall filen modalias existerar i sysfs, betyder det att drivrutinen stödjer enheten och kan tala till den direkt, men ifall den inte har aliaset är det en bugg i drivrutinen. Ladda drivrutinen utan hjälp av Udev och anta att problemet löses framöver.

Ifall det inte finns en modalias fil i den korrekta mappen, innebär detta att kernel-utvecklarna ännu inte implementerat modalias stöd för denna bus-typ. För Linux-5.5.3 är det fallet för ISA bussar. Utgå ifrån att denna typ av problem löses i framtida kernelversioner.

Udev är inte tänkt att ladda “wrapper” drivrutiner såsom snd-pcm-oss, eller ickehårdvaru drivrutiner som loop.

7.3.3.2. En kernel-modul laddas inte automatiskt, och Udev är inte tänkt att ladda den

Ifall “wrapper”-modulen endast förbättrar funktionaliteten tillgängligjord av en annan modul (t.ex snd-pcm-oss förbättrar snd-pcm genom att göra ljudkorten tillgängliga för OSS applikationer), konfigurera då **modprobe** till att ladda wrappern efter att Udev laddat den wrappade modulen. För att göra detta, lägg till en “softdep”-rad till /etc /modprobe.d / filnamn.conf. Till exempel:

```
softdep snd-pcm post: snd-pcm-oss
```

Notera att kommandot “softdep” även tillåter pre: dependencies, eller en mix av både pre: och post:. Se manualsidan modprobe.d(5) för mer information vad gäller “softdep” syntax och funktionalitet.

Ifall modulen ifråga inte är en wrapper utan användbar i sig självt, konfigurera då bootskriptet **modules** så att det laddar denna modul vid uppstart. För att göra detta, lägg till modulnamnet i en separat rad i filen /etc /sysconfig /modules. Detta fungerar för wrapper moduler också, men är inte att betrakta som en optimal lösning.

7.3.3.3. Udev laddar oönskade moduler

Antingen bygg inte modulen, eller svartlista den i en /etc /modprobe.d /blacklist.conf fil. Detta görs i kommandot nedan för modulen “forte”.

```
blacklist forte
```

Svartlistade moduler kan fortfarande laddas manuellt, med hjälp av kommandot **modprobe**.

7.3.3.4. Udev skapar en enhet på ett inkorrekt vis, eller skapar en felaktig symlink

Vanligtvis inträffar detta då en regel oförväntat matchar en enhet. En illa skriven regel kan exempelvis matcha både en önskad SCSI-disk och den motsvarande men oönskade generiska SCSI enheten. Lösningen är att hitta den felande regeln och göra den mer specifik. Används kommandot **udevadm info** för detta syfte.

7.3.3.5. Udev-reglerna fungerar på ett opålitligt sätt och vis

Detta kan vara ytterligare en manifestation av det föregående problemet. Ifall inte, och din regel använder sysfs-attribut, kan det vara ett kernel timing problem, som kommer fixas i senare kernelversioner. Tillsvidare kan du arbeta runt det genom att skapa en regel som väntar på det används sysfs attributet, samt lägga till denna regel i filen /etc /udev /rules.d /10-wait_for_sysfs.rules (skapa denna fil om den ej existerar).

7.3.3.6. Udev skapar inte enheten

Ytterligare text utgår ifrån att drivrutinen byggs in i kerneln statiskt eller redan laddad som en modul, och att du redan har kollat att Udev skapar enheter som namngivs korrekt.

Udev har inte den information vilken krävs för att skapa en enhetsnod, ifall inte en kernel-drivrutin exporterar sin data till Sysfs. Detta är vanligast med tredjeparts drivrutiner från utanför kernel trädet. Skapa en statisk enhetsnod i /lib /udev / devices med de lämpliga major/minor numren (se filen devices.txt inuti kerneldokumentationen eller dokumentationen tillhandahållen av utvecklaren av tredjeparts drivrutinen). Den statiska enhetsnoden kommer kopieras till /dev av **udev**.

7.3.3.7. Namn-ordning för enheter förändras vid omstarter

Detta beror på att Udev, till sin design, hanterar och laddar uevents och moduler parallellt, och därför i en oförutsägbar ordning. Detta kommer aldrig att “fixas”. Du bör inte förlita dig på att enhetsnamn är stabila över tid. Skapa istället dina egna regler som skapar symlänkar med stabila namn baserad på några stabila attribut hos enheten, som t.ex ett serienummer eller output från diverse *_id verktyg installerade av Udev. Referera till sektion 7.4 och 7.5 för exempel.

7.3.4. Användbara referenser

Användbar dokumentation finns även tillgängligt på följande webbsidor:

- En Userspace Implementation av `devfs` *http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf*
- Filesystemet sysfs *http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf*

7.4. Att hantera enheter

7.4.1. Nätverks-enheter

Som standard namnger Udev nätverksenheter enligt Firmware/BIOS data eller fysisk karaktär såsom bus eller MAC-adress. Anledningen till denna namngivningskonvention är att säkerställa att nätverksenheterna namnges på ett konsistent vis och inte baserat på t.ex den tid då nätverskortet upptäcktes. På en dator med två nätverkskort kan det ena namnges till eth0 och det andra till eth1. Numrena kan skifta efter t.ex en omstart.

Med den nya namngivnings-konventionen kommer nätverksenheter döpas till enp5s0, wlp3s0 eller liknande. Ifall denna namngivnings-konventionen inte är önskvärd, kan den traditionella konventionen, eller en anpassad, implementeras.

7.4.1.1. Avaktivera Persistent Naming i Kernel Command Line

Den traditionella namngivningskonventionen, som använder eth0, eth1, etc, kan återinrättas genom att net.ifnames=0 läggs till via kernel kommandoraden. Detta lämpar sig bäst för de system som endast han en ethernet enhet av samma typ. Laptops har ofta multipla ethernet uppkopplingar som är namngivna eth0 wlan0 och kan också använda sig av denna metod. The command line is passed in the GRUB configuration file. Se sektion 8.4.4.

7.4.1.2. Att skapa regler för Udev

Namngivningskonventionen kan anpassas genom skapandet av Udev regler. Ett skript är inkluderat, vilket genererar de initiala reglerna. Generera dessa regler genom att köra skriptet:

```
bash /lib/udev/init-net-rules.sh
```

Inspektera nu filen `/etc/udev/rules.d/70-persistent-net.rules` för att ta reda på vilket namn som tillskrivits vilken enhet.

```
cat /etc/udev/rules.d/70-persistent-net.rules
```

Kommentar

I vissa fall, som då MAC adresser manuellet tillskrivits åt ett nätverkskort eller i en virtuell miljö såsom Qemu eller Xen, kan det vara så att filen med nätverksregler inte genererats eftersom adresser inte tillskrivs enheter på ett sammanhängande vis. i dessa fall, kan inte denna metod användas.

Filen börjar med ett komment block efterföljt av två rader för varje NIC. Den första raden för varje NIC är en kommenterad beskrivning som visar hårdvaru IDn (t.ex dess PCI producent och enhets IDn, ifall det är ett PCI kort), tillsammans med drivrutinen inom paranteser ifal drivrutinen kan hittas. Varken hårdvaru ID eller drivrutin används för att avgöra vilket namn ett gränssnitt ska tillskrivas; denna information är endast för referens. Den andra raden är Udev regeln som matchar NICet och faktiskt tillskriver det ett namn.

Alla Udev regler består av ett antal nycklar, separerade från varandra med komman och whitespaces. Denna regels nycklar samt vad de gör återfinns nedan:

- `SUBSYSTEM=="net"` - Specifierar för Udev att ignorera enheter som inte är nätverkskort.
- `ACTION=="add"` - Specifierar för Udev att ignorera denna regel för uevent som inte är ett “add”(“remove” och “change” uevents inträffar också, men behöver inte ge nätverks gränssnitt nya namn).
- `DRIVERS=="?*"` - Existerar så att Udev kommer ignorera VLAN eller bridge sub-interfaces (eftersom dessa sub-interfaces inte har drivrutiner. Dessa sub-interfaces ignoreras eftersom namnet som skulle tillskrivas dem skulle kollidera med deras föräldra enheter.
- `ATTR{address}` - Denna nyckels värde är NICets MAC adress.
- `ATTR{type}=="1"` - Säkerställer att regeln endast matchar det huvudsakliga gränssnittet när det kommer till vissa trådlösa drivrutiner, vilka skapar multipla virtuella gränssnitt. De sekundära gränssnitten hoppas över pga samma anledning som VLAN och sub-interfaces hoppas över: namn-kollisioner
- `NAME` - Denna nyckels värde är det namn vilket Udev kommer tillskriva gränssnittet.

Värdet inuti `NAME` är den viktigaste aspekten. Säkerställ att du vet vilka namn som tillskrivits dina nätverkskort innan du fortsätter vidare, och se till att använda värdet `NAME` då du nedan skapar konfigurationsfilerna.

7.4.2. CD-ROM symlänkar

En del mjukvara som du kanske vill installera senare (t.ex diverse mediaspelare) förväntar sig att /dev /cdrom och /dev /dvd symlänkarna existerar, och att dessa pekar till en CD-ROM eller DVD-ROM enhet. Det kan även vara smidigt att placera referenser till dessa symlänkar i /etc /fstab. Udev inkluderar ett skript som kommer generera regel-filer för att skapa dessa symlänkar åt dig, beroende på funktionaliteten hos varje enhet, men du behöver välja mellan två olika typer av tillstånd för hantering som du önskar att skriptet använder.

Det första tillståndet innebär att skriptet använder “by-path” (används av USB och FireWire enheter), där reglerna som skapas beror på den fysiska sökvägen till en CD eller DVD enhet. Det andra tillståndet fungerar enligt “by-id” (standard för IDE och SCSI enheter), där reglerna som skapas beror på identifikationssträngar sparade i CD eller DVD enheten självt. Sökvägen bestäms av Udev’s **path_id** skript, och identifikationssträngarna läses från hårdvaran med programmet **ata_id e**ller **scsi_id**, beroende på vilken typ av enhet du har.

Det finns fördelar med båda alternativen. Det korrekta valet beror på vilka typer av enhetsförändringar som kommer ske. Om du förväntar dig att den fysiska sökvägen till enheten (alltså portar eller var den pluggas in i datorn) kommer ändras, då bör du använda “by-id” alternativet. Om du å andra sidan förväntar dig att enhetens identifikation kanske kommer att ändras, om den t.ex ersätts av en annan enhet med samma förmågor och som pluggas in i samma plats på datorn, du bör du använda “by-path” alternativet.

Om du förväntar dig att båda typer av förändringar kommer att ske, välj då det alternativ som du tror kommer vara det som för det mesta är mest användbart.

Viktigt

Externa enheter (t.ex en USB-inkopplad CD-enhet) bör inte använda “by-path” persistence, eftersom varje gång som enheten pluggas in i en ny extern port, dess fysiska sökväg kommer att ändras. Alla externt inkopplade enheter kommer lida av detta problem ifall du skriver Udev regler som identifierar enheter med hjälp av deras fysiska sökvägar; problemet är inte begränsat till CD och DVD-enheter

Om du önskar visa värdena som Udev skripten kommer att använda, hitta då, för den korrekta CD-ROM enheten, den motsvarande mappen under /sys (detta kan t.ex vara /sys /block /hdd) och kör ett kommando liknande det här nedan:

```
udevadm test /sys/block/hdd
```

Analysera raderna som innehåller output för de diverse *_id programmen. Alternativet “by-id” kommer använda värdet ID_SERIAL om det existerar och inte är tomt, annars kommer det använda en kombination av ID_MODEL och ID_REV ISION. Alternativet “by-path” kommer använda värdet ID_PATH.

Om standardalternativet inte är lämplig för dina omständigheter, då kan följande modifikationer göras i filen /etc /udev / rules.d/ 83-cdrom-symlinks.rules. “mode” i texten nedan representerar var “by-id” eller “by-path” ska stå.

```
sed -i -e 's/"write_cd_rules"/"write_cd_rules mode"/' \
    /etc/udev/rules.d/83-cdrom-symlinks.rules
```

Notera att det inte är nödvändigt att skapa regelfilerna eller symlänkarna här och nu, eftersom du har bind-monterat värdens /dev mapp in i LFS systemet, och vi antar att symlänkare existerar på värden. Reglerna och symlänkarna kommer att skapas första gången som du startar upp LFS systemet.

Ifall du däremot har multipla CD-ROM enheter, då kan det leda till att symlänkarna som skapas kommer peka till andra enheter än vad de pekar till på din värd, eftersom enheter inte upptäcks i en förutsägbar ordning. De symlänkar som skapas vid uppstart kommer vara stabila för LFS, så detta är endast ett problem ifall värd och LFS måste ha samma pekare. Ifall detta är nödvändigt, redigera då, efter uppstart, filen /etc /udev /rules.d /70-persistent-cd.rules.

7.4.3. Att hantera enhets-dubletter

Som förklarades i sektion 7.3, är ordningen i vilken enheter med samma funktion framträder i /dev slumpmässig. Ifall du t.ex har en USB-kamera och en TV-tuner, kommer /dev /video0 ibland peka till kameran och ibland till tunern. För alla typer av hårdvara, bortsett ljudkort och nätverkskort, kan detta fixas genom att skapa Udev-regler för anpassade persistent symlänkar. För mer information om detta vad gäller nätverkskort, se sektion 7.5. Konfiguration för ljudkort återfinns i boken BLFS.

För var och en av dina enheter som kan tänkas få detta problem (även ifall problemet inte existerar i din nuvarande distro), hitta den motsvarande mappen under /sys /class eller /sys /block. För video-enheter kan detta vara /sys /class /video4linux /videoX. List ut vilka attribut som på ett unikt vis identiferar enheten (vanligtvis attribut såsom producent, produkt-id, och/eller serienummer):

```
udevadm info -a -p /sys/class/video4linux/video0
```

Skriv sedan reglerna som skapar symlänkarna. T.ex:

```
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 blir att /dev /video0 även i fortsättningen pekar emot kameran eller TV-tunern på ett slumpmässigt vis (och till följd av detta aldrig bör användas direkt), men att det finns symlänkar /dev /tvtuner och /dev /webcam som alltid pekar till den korrekta enheten.

7.5. Generell nätverks-konfiguration

7.5.1. Skapa Network Interface Configuration Files

Vilket gränssnitt som tas upp och ned av nätverks skriptet beror vanligtvis på filerna i /etc /sysconfig/. Denna mapp bör därför innehålla en fil för varje gränssnitt som ska konfigureras, såsom ifconfig.xyz, där “xyz” beskriver nätverskkortet. Gränssnittets namn (t.ex eth0) är vanligtvis lämpligt. Inuti denna fil finns attribut för detta gränssnitt, såsom dess IP-ad ress(er), subnet masks, och så vidare. Det är nödvändigt att filen heter “ifconfig”.

Kommentar

Om proceduren i föregående sektion inte användes, kommer Udev att tillskriva nätverkskort dess namn baserat på systemets fysiska karaktär såsom enp2s1. Om du inte är säker på vilket ditt gränssnitts namn är, kan du alltid köra **ip link** eller **ls /sys/class/net** efter att du har startat upp systemet.

Följande kommando skapar en exempel fil, med en statisk ip-adress, för enheten eth0.

```
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
```

De värden som är markerade med italics måste i varje fil ändras, så att de korrekt matchar systemet.

Ifall variabeln ONBOOT är satt till “yes” kommer system V nätverks skriptet att sätta upp nätverkskortet(NIC) under själva uppstarten av systemet. Ifall ONBOOT är satt till något annat än “yes”, kommer NIC att ignoreras av skriptet och inte sättas upp automatiskt. Gränssnittet kan startas och stoppas med kommandona **ifup** och **ifdown**.

Variabeln IFACE definierar gränssnittets namn, t.ex eth0. Det krävs av alla konfigurations-filer för nätverksenheter. Filnamnsförlängningen måste matcha detta värde.

Variabeln SERVICE definierar metoden som används för att framskaffa IP-adressen. Paketet LFS-Bootscripts använder sig av ett modulärt IP-assignment format, och att skapa ytterligare filer i /lib/services/ mappen möjliggör andra IP-assig nment metoder. Detta används vanligtvis för DHCP, vilket hanteras i boken BLFS.

Variabeln GATEWAY bör innehålla standard gatewayens IP-adress, ifall en gateway finns tillgänglig. Ifall det inte gör det, kommentera då ut denna variabel.

Variabeln PREFIX innehåller numret av bits som används i subnätet. Varje octal i en IP-adress är 8 bits. Ifall subnäts netmasken är 255.255.255.0, då använder den de första tre octalen (24 bits) för att specifiera nätverks-numret. Ifall netmasken är 255.255.255.240, då använder den de första 28 bitsen. Prefix längre än 24 bits används vanligtvis av DSL och kabel-baserades ISPs. i detta exempel (PREFIX=24) är netmasken 255.255.255.0. Justera variabeln PREFIX till att motsvara ditt nuvarande subnät. Ifall denna variabel inte definieras, är dess standardvärde 24.

Referera till man sidan för **ifup** om du vill ha mer information.

7.5.2. Skapa filen /etc/resolv.conf

Systemet kommer behöva funktionalitet för DNS, så att IP-adresser kan omvandlas till internet domän namn (och vice-versa). Det implementeras bäst genom att man placerar IP-adressen för DNS-servern, tillgänglig via ISP eller nätverks-administratören, i filen /etc /resolv.conf. Skapa denna fil:

```
cat > /etc/resolv.conf << "EOF"
# Begin /etc/resolv.conf

domain <Your Domain Name>
nameserver <IP address of your primary nameserver>
nameserver <IP address of your secondary nameserver>

# End /etc/resolv.conf
EOF
```

Uttrycket “domain” kan utelämnas eller ersättas med ett “search”-uttryck. För mer information, referera till man sidan för resolv.conf.

Ersätt <IP address of your primary nameserver> med IP-adressen till den DNS-server som lämpar sig bäst för systemet. Det finns ofta mer än en entrie vad gäller DNS-server (för att uppfylla krav på redundans etc.). Om du bara behöver eller vill ha en DNS-server, radera den andra NS-raden från filen. IP-adressen kan även vara en router på det lokala nätverket.

Kommentar

Googles offentliga IPv4 DNS adresser är 8.8.8.8 och 8.8.4.4.

7.5.3. Konfigurera systemets värdnamn

Vid uppstart av systemet används filen /etc /hostname för att etablera systemets värdnamn.
Skapa denna fil och specifiera ett värdnamn genom att köra:

```
echo "<lfs>" > /etc/hostname
```

<lfs> behöver ersättas med namnet på själva datorn. Skriv inte in FQDN här. Den informationen placeras i filen /etc/hosts

7.5.4. Anpassa filen /etc/hosts

Bestäm IP-adress, fully-qualified domain name (FQDN), samt möjliga alias som ska användas i /etc//hosts.
Syntaxen är:

```
IP_address myhost.example.org aliases
```

Ifall datorn inte är tänkt att vara synlig för Internet (det finns en registrerad domän och ett giltigt block av IP-adresser – de flesta användare har inte detta), säkerställ då att IP-adressen befinner sig i nätverkets privata IP-adress spektrum. Giltiga spektrum är:

```
Private Network Address Range      Normal Prefix
10.0.0.1 - 10.255.255.254           8
172.x.0.1 - 172.x.255.254           16
192.168.y.1 - 192.168.y.254         24
```

“x” kan vara ett tal i spektrumet 16-31. “y” kan vara ett tal i spektrumet 0-255.

En giltig IP-adress kan vara 192.168.1.1. En giltig FQDN för denna IP-adress kan vara lfs.example.org.

Även ifall ett nätverskort inte använda, är en giltigt FQDN ofta nödvändig. Detta är nödvändigt för att vissa program ska kunna fungera korrekt.

Skapa filen `/etc/hosts` genom att köra:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 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
```

Värdena *<192.168.1.1>*, *<FQDN>* och *<HOSTNAME>* behöver ändras för specifika användningsområden eller krav. Aliasen kan utelämnas.

7.6. System V Bootscript användning och konfiguration

7.6.1. Hur fungerar Bootscripts för System V?

Linux använder en särskilt uppstarts funktionalitet kallad SysVinit som baseras på ett koncept kallat *run-levels*. Hur detta Fungerar kan skilja sig åt emellan olika system, så bara för att saker och ting fungerade i en Linux distro, är det inte Säkert att samma saker fungerar i LFS. LFS har ett eget sätt att göra saker på, men respekterar generella standarder.

SysVinit (vilket kommer refereras till som “init”) använder en run-levels schematik. Det finns sju (numrerade 0-6) run-Levels (det finns faktiskt fler, men de är för särskilda omständigheter och används vanligtvis inte. Se init(8) för mer info). Varje av dessa motsvarar en typ av handlingar som datorn är tänkt att genomföra vid uppstart. Standard run-level är 3. Nedan är beskrivningar för run-levels:

```
0: halt the computer
1: single-user mode
2: multi-user mode without networking
3: multi-user mode with networking
4: reserved for customization, otherwise does the same as 3
5: same as 4, it is usually used for GUI login (like X's xdm or KDE's kdm)
6: reboot the computer
```

7.6.2. Konfigurera Sysvinit

Under själva initialiseringen av kerneln, kommer det första program som körs att antingen specifieras av kommandoraden eller, som standard, av **init**. **Init** läser initialiserings-filen /etc/inittab. Skapa därför denna fil:

```
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:S016:once:/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 genomgång av denna initialiserings-fil återfinns i man sidan för inittab. För LFS är det centrala kommandot **re**. Filen ovan kommer att instruera **re** att köra alla skript som börjar med S i mappen /etc/rc.d/rcS.d vilket sedan följs av alla skript som börjar med S i mappen /etc/rc.d/rc?.d (där frågetecknet specifieras av värdet initdefault.

För enkelhetens skull, läser skriptet **re** ett bibliotek av funktioner i /lib/lsb/init-functions. Detta bibliotek läser även, som tillval, en konfigurations-fil /etc/sysconfig/rc.site. System konfigurations fil-parametrar vilka beskrivs i följande sektioner kan alternativt placeras i denna fil, vilket möjliggör konsolidering av alla system-parametrar till en plats.

Som en felsöknings hjälp, loggar funktionens skript även all output till /run /var /bootlog. Eftersom mappen /run är en tmpfs så sparas dock inte denna fil emellan omstarter. Vid nedstängning av systemet så skrivs dock innehållet i denna til till filen /var/log/boot.log

7.6.2.1. Att ändra Run Levels

Att ändra run-levels görs med **init <runlevel>,** där <runlevel> är den run-level som ska ändras. För att t.ex starta om datorn, kan en användare åberopa kommandot **init 6**, vilket är ett alias för **reboot**. På samma sätt är **init 0** ett alias för Att föra systemet till halt.

Det finns ett antal mappar under /etc /rc.d vilka ser ut som rc?.d (där ? Är run-level) och rcsysinit.d. Alla dessa innehåller ett antal symboliska länkar. Vissa börjar med ett “K”, andra med “S”, och alla har två nummer som efterföljer denna inledande bokstav. “K” betyder att stoppa (kill) en tjänst och “S” att starta en tjänst. Numren bestämmer ordningen i vilken skripten ska köras, från 00 till 99 där ett lägre nummer innebär tidigare exekvering. Då **init** byter från en run-level till en annan, stoppas eller startas de lämpliga tjänsterna beroende på vilken run-level som valts.

De riktiga skripten återfinns i /etc /rc.d /init.d. De utför det verkliga arbetet, och alla symlänkar pekar till dem. “K” och “S” länkar pekar alla till samma skript i /etc /rc.d /init.d. Anledningen till detta är att skripten kan kallas med med olika parametrar som start, stop, restart, reload och status. När en “K” länk upptäcks, körs det korrekta skriptet med argumentet stop. När en “S” länk upptäcks, körs det korrekta skriptet med argumentet start.

Det finns ett undantag vad gäller detta. Länkar som börjar med ett “S” i mapparna rc0.d och rc6.d kommer inte leda till att något startas. De kommer att kallas med argumentet stop för att stoppa något. Logiken bakom detta är att då en användare tänker starta om eller halta systemet, behöver inget startas. Systemet behöver då endast stoppas.

Detta är beskrivningar för vad argumenten får skripten att göra:

start
: Tjänsten startas.

stop
: Tjänsten stoppas.

restart
: Tjänsten stoppas först, och startas sedan upp igen.

reload
: Tjänstens konfiguration uppdateras. Används efter att en konfigurationsfil har modifierats och tjänsten inte behöver startas om.

status
: Rapporterar ifall tjänsten är aktiv, samt under vilka PIDs den är aktiv.

Modifiera gärna hur uppstarts-processen fungerar (det är i slutändan ditt LFS-system). Filerna som visas här är exempel på hur det kan göras.

7.6.3. Udev Bootscripts

/etc/rc.d/init.d/udev initskriptet startar **udev**, triggar “coldplug” enheter vilka redan skapats av kerneln och väntar på att regler ska slutföras. Skriptet avmonterar även uevent hanteraren från standard /sbin /hotplug. Detta görs eftersom kerneln inte längre behöver kalla ett externt bibliotek. **udevd** kommer istället lyssna på en netlink socket efter uevent som kerneln skapar.

/etc/rc.d/init.d/udev_retry initskriptet ser till att åter-trigga händelser för subsystem vars regler kanske är beroende av filsystem vilka inte monteras innan skriptet **mountfs** körs (särskilt /usr och /var kan leda till detta). Detta skript körs efter **mountfs** skriptet, så dessa regler borde (om åter-triggade) lyckas andra gången de körs. Skriptet konfigureras från filen /etc /sysconfig /udev_retry. Alla ord i denna fil, bortsett från kommentarer, betrakts som subsystem namn att trigga. För att hitta subsystemet för en enhet, använd **udevadm info --attribute-walk <device>,** där <device> är en absolut sökväg i /dev eller /sys (såsom /dev/sr0 eller /sys/class/rtc.

För information vad gäller udev och module laddning, se sektion 7.3.2.3.

7.6.4. Konfigurera systemets klocka

Skriptet **setclock** läser tiden från hårdvaru-klockan, även kallad BIOS eller Complementary Metal Oxide Semiconductor (CMOS) klockan. Ifall hårdvaru-klockan är satt till UTC, kommer detta skript konvertera hårdvaru-klockans tid till den lokala tiden genom att använda filen /etc /localtime (vilken specifierar för **hwclock** vilken tidszon som användaren befinner sig i). Det går inte att avgöra ifall hårvaru-klockan är satt till UTC, så detta behöver konfigurerars manuellt.

Setclock körs via udev när kerneln vid uppstart upptäcker denna hårdvaru förmåga. Det kan även köras manuellt med parametern stop för att spara systemtiden till CMOS-klockan.

Ifall du inte minns vare sig hårdvaru-klockan är satt till UTC, ta då reda på detta genom att köra **hwclock --localtime --show**

```
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
```

En bra källa för att lära sig om hur LFS hanterar tid är: http://www.linuxfromscratch.org/hints/downloads/files/time.txt

Kommentar

Parametrarna CLOCKPARAMS och UTC kan även definieras i filen /etc/sysconfig/rc.site

7.6.5. Konfigurera Linux konsol

Denna sektion diskuterar hur man konfigurerar **console** bootskriptet som sätter upp tangentbords kartan, konsol fonten, samt konsolens kernel log nivå. Ifall icke-ASCII karaktärer (t.ex symbolen för det Brittiska punder eller euron) inte kommer att användas och tangentbordet är ett Amerikanskt sådant, kan man hoppa över mycket av denna sektion. Utan konfigurations-filen (eller motsvarande inställningar i rc.site) kommer **console** bootskriptet inte göra något.

console skriptet läser filen /etc/sysconfig/console för konfigurations information. Avgör vilken keymap och skärm font som ska användas. Diverse språk-specifika HOWTOs kan också vara av nytta, se http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html. Om du ändå är osäker på hur att genomföra detta, kolla i /usr/share/keymaps och /usr/share/consolefonts mapparna för giltiga keymaps och skärm fonts. Läs loadkeys(1) och setfont(8) manual sidor för att avgöra vilka de korrekta argumenten är för dessa program.

Filen /etc/sysconfig/console bör innehålla rader i formen: VARIABLE=”value”. Följande variable igenkänns:

LOGLEVEL

Specifierar loggnings-nivå för kernel meddelanden, som bestämt av **dmesg**. Giltiga nivåer är “1” (inga meddelanden) till “8”. Standardnivån är “7”.

KEYMAP

Denna variabel specifierar argumenten för programmet **loadkeys**, vilket vanligtvis är namnet för önskad keymap. ifall denna variabel inte är specifierad, kommer bootskriptet inte köra **loadkeys**, och standard kernel keymap kommer att användas. Notera att ett antal keymaps har multipla versioner med samma namn (cz och dess varianter i qwerty/ och qwertz/, es i olpc/ och qwerty/, samt trf i fgGlod/ och qwerty/). i dessa fall för föräldramappen även specifieras för att säkerställa att korrekt keymap används.

KEYMAP_CORRECTIONS

Denna (sällan använda) variabel specifierar argumenten för det andra anropet till **loadkeys**. Detta är användbart ifall stock keymapen inte är helt och hållet tillfredställande och mindre justeringar måste göras (t.ex att inkludera euro-symbolen till en keymap som vanligtvis inte har den, sätt denna variabel till “euro2”).

FONT

Denna variabel specifierar argumenten för programmet **setfont**. Detta inkluderar vanligtvis font-namnet, “-m”, samt namnet på applikationens character map att ladda. Att t.ex ladda fonten “lat1-16” tillsammans med applikations karaktärs mapen “8859-1” (då denna är lämplig i USA), sätt denna variabel till “lat1-16 -m 8859-1”. i tillståndet UTF-8, använder kerneln applikations mapen för omvandling av 8-bit nyckel koder till UTF-8, och därför skall argumentet för parametern “-m” sättas till den encoding som key codes har i keymapen.

UNICODE

Sätt denna variabel till “1”, “yes” eller “true” för att placera konsolen i UTF-8 tillstånd. Detta är användbart i UTF-8 baserade locales men skadligt annars.

LEGACY_CHARSET

För många tangentbords layouts, finns det ingen stock Unicode keymap i Kbd paketet. **console** bootskriptet kommer konvertera en tillgänglig keymap till UTF-8 ifall denna variabel är satt till encodingen för den tillgängliga ej-UTF-8 keymapen.

Några exempel:

- För en ej-Unicode setup, behövs vanligtvis endast KEYMAP och FONT variablerna. T.ex för en Polsk setup skulle man använda:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# End /etc/sysconfig/console
EOF
```

- Som nämndes ovan är det ibland nödvändigt att justera en stock keymap. Följande exempel lägger till Euro-symbolen till keymapen för Tyska:

```
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
```

- Följande är ett Unicode-aktivera exempel för Bulgariska, där en UTF-8 keymap existerar:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# End /etc/sysconfig/console
EOF
```

- Till följd av användningen av en 512-glyph LatArCyrHeb-16 font i det föregående exemplet, är ljusa färger inte längre tillgänliga via Linux konsolen ifall inte en framebuffer används. Om man önskar använda ljusa färger utan en framebuffer och kan leva utan karaktärer som inte tillhör det valda språket, är det fortfarande möjligt genom att använda en språk-specifik 256-glyph font, som illustreras nedan:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# End /etc/sysconfig/console
EOF
```

- Följande exempel illustrerar keymap autoconversion från ISO-8859-15 till UTF-8 och aktiverandet av döda tangenter i Unicode tillstånd:

```
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
```

- Vissa keymaps har döda tangenter (tangenter vilka inte producerar en karaktär själva, men istället modifierar karaktären som skrivs in av nästa tangent) eller definierar sammansättningsregler (t.ex: “tryck ctrl+A E för att få Æ”). Linux 5.3.3 tolkar döda tangenter och sammansättningsregler i keymapen korrekt endast då source-karaktärerna som ska sammansättas inte är multibyte. Denna begränsning påverkar inte keymaps för Europeiska språk, eftersom accenter där läggs till för icke-accentuerade ASCII karaktärer, eller så är två ASCII karaktärer sammanfogade. i UTF-8 leder det dock till problem för exempelvis Grekiska, där man ibland måste accentuera bokstaven “alpha”. Lösningen här är att antingen undvika användandet av UTF-8, eller att installera X window systemet som inte begränsas på detta vis när det gäller dess input hantering.

- För Kinesiska, Japanska, Koreanska samt ett antal andra språk, kan inte Linux konsolen konfigureras för att visa de nödvändiga karaktärerna. Användare som behöver sådana språk bör installera X Window System, fonts som täcker de nödvändiga karaktärs-spektrumen, samt den korrekta input-metoden (t.ex SCIM, som stödjer ett stort antal språk.

Kommentar

Filen /etc/sysconfig/console kontrollerar endast Linux text konsol lokalisering. Den har inget att göra med att Sätta den korrekta tangentbords layouten eller terminal fonten i X Window System, med ssh sessioner eller Med en seriell konsol. i sådana situationer, gäller inte begränsningarna som nämns i de två sista punkterna ovan.

7.6.6. Skapa filer vid uppstart

Det är ibland önskvärt att skapa filer vid uppstart. T.ex mappen /tmp/.ICE-unix. Detta kan göras genom att man skapar en entry i konfigurations-skriptet /etc/sysconfig/createfiles. Denna fils format är inbäddat i kommentarerna i standard konfigurations-filen.

7.6.7. Konfigurera skriptet sysklogd

Skriptet sysklogd åkallar programmet **syslogd** som del av initialiseringen av System V. Alternativet “-m 0” stänger av den periodiska tidsstämpeln vilkem **syslogd** skriver till logfiler som standard var 20e minut. Ifall du önskar stänga av denna periodiska stämpel, redigera /etc/sysconfig/rc.site och definiera variabeln SYSKLOGD_PARMS till det önskade värdet. För att t.ex radera alla parametrar, sätt variabeln till ett null värde:

```
SYSKLOGD_PARMS=
```

Se **`man syslogd`** för fler alternativ.

7.6.8. Filen rc.site

Den som tillval tillgängliga filen /etc/sysconfig/rc.site innehåller inställningar som automatiskt sätts för varje System V boot skript. Alternativt kan den sätta värdena specifierade i filerna hostname, console, och clock i mappen /etc/sysconfig/. Ifall de relaterade variablerna finns tillgängliga i både dessa separata filer samt i rc.site, har värdena i de skript-specifika filerna företräde.

rc.site innehåller även parametrar som kan anpassa andra aspekter av uppstarts-processen. Att sätta variabeln IPROMPT kommer aktivera selektivt körande av bootskript. Andra alternativ beskrivs i fil-kommentarerna. Standard versionern av filen ser ut som nedan:

```
# 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@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 seet 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=
```

7.6.8.1. Anpassa skripten för uppstart och nedstängning

Boot-skripten för LFS startar upp och stänger ned systemet på ett relativt effektivt sätt, men det finns ett antal förändringar vilka du kan göra i filen rc.site för att förbättra hastigheten ännu mer och för att justera meddelande enligt dina egna önskemål. För att göra detta, justera inställningarna i /etc/sysconfig/rc.site:

- Då boot-skriptet udev körs, åberopas **udev settle** som tar ett tag att slutföra. Denna tid kanske inte är nödvändig vilket beror på enheterna som finns tillgängliga på systemet. Om du endast har simpla partitioner och ett enstaka nätverks-kort, behöver uppstarts-processen antagligen inte vänta på detta kommando. För att hoppa över det, sätt variabeln: OMIT_UDEV_SETTLE=y
- Boot-skriptet udev_retry kör även, som standard, **udev settle**. Detta kommando krävs endast då som standard ifall mappen /var monteras separat. Detta beror på att klockan behöver filen /var/lib/hwclock/adjtime. Andra anpassningar kan också behöva vänta på udev att slutföras, men för många installationer krävs inte detta. Skippa kommandot med: OMIT_UDEV_RETRY_SETTLE=y
- Som standard är filsystems dubbelkollarna ljudlösa. Detta kan framstå som en försening under uppstart. För att sätta på **fsck** output, sätt variabeln: VERBOSE_FSCK=y
- Vid omstart kanske du vill hoppa över filsystems kollen vilke **fsck** genomför helt och hållet. För att göra detta skapa filen /fastboot eller starta om systemet med kommandot **/sbin/shutdown -f -r now**. Å andra sidan, kan du tvinga alla filsystem att kollas genom att skapa /forcefsck eller genom att köra **shutdown** med parametern -F istället för -f.

Att sätta variabeln FASTBOOT=y kommer deaktivera **fsck** vid uppstarts-processen tills dess den raderas. Detta är inte att rekommendera som permanent hantering.

- Vanligtvis raderas alla filer i mappen /tmp vid uppstart. Beroende på hur många mappar samt filer som finns där, kan detta leda till en betydande fördröjning av uppstarten. För att hoppa över raderandet av dessa filer sätt variabeln: SKIPTMPCLEAN=y
- Vid nedstängning skickar **init** en signal till varje program som det har startat (t.ex agetty), väntar under en specifierad tid (standard 3 sekunder), och skickar sedan varje process en KILL signal och väntar sedan åter. Denna process repeteras i skriptet **sendsignals** för varje process som inte stängts ned av det egna skriptet. Fördröjningen för **init** kan specifieras genom att en parameter skickas med. För att till exempel ta bort fördröjningen, skicka -t0 vid omstart eller nedstängning (t.ex **/sbin/shutdown -t0 -r now**). Fördröjningen för **sendsignals** skriptet kan hoppas över genom att man sätter parameter KILLDELAY=0.

7.7. Filer för bash-skalets uppstart

Skalprogrammet **/bin/bash** (hädanefter refereat till som “skalet”) använder sig av en samling uppstarts filer för att hjälpa till att skapa en miljö i vilket det kan köras. Varje fil har ett specifikt användningsområde och kan påverka login samt interaktiva miljöer på olika sätt. Filerna i /etc tillhandahåller globala inställningar. Ifall en motsvarande fil existerar i home mappen, så kan detta leda till att den filen prioriteras över de globala inställningarna.

Ett interaktivt skal startas efter en framgångsrik inloggning, användande **/bin/login**, genom att läsa filen /etc/passwd. Ett interaktivt skal startas för kommandoraden. Ett icke-interaktivt skal är vanligtvis närvarande då ett skal-skript körs. Det är icke-interaktivt eftersom det processar ett skript och inte väntar för användar-input emellan skriptets kommandon.

Filerna /etc/profile och ~/.bash_profile läses då skalet åkallas som ett interaktivt inloggnings-skal.

Den grundläggande /etc/profile nedan sätter ett antal miljö variabler som krävs för systemets språk-stöd. Att specifiera dem korrekt resulterar i:

- Program output översatt till systemets språk
- Korrekt klassifikation av karaktärer till bokstäver, siffror eller andra klasser. Detta är nödvändigt för att **bash** på ett korrekt sätt ska kunna acceptera icke-ASCII karaktärer hos kommandorader med icke-Engelska locales.
- Korrekt alfabetisk sortering för landet
- Lämplig pappers-storlek
- Korrekt formatering av monetära, tids relaterade, och datum värden

Ersätt <11> nedan med en två-bokstavs kod för det önskade språket (t.ex “en”) och <CC> med två-bokstavs koden för det korrekta landet (t.ex “GB”). <charmap> bör ersättas med korrekt canonical charmap för den valda locale. Extra modifierare som “@euro” kan också finnas tillgängliga.

En lista med alla locales som stödjs av glibc kan erhållas genom att köra:

```
locale -a
```

Charmaps kan ha ett antal alias, t.ex “ISO-8859-1” refereras även till som “iso8859-1” och “iso88591”. Vissa applikationer kan inte hantera de diverse synonymerna korrekt (kräver t.ex att “UTF-8 skrivs som “UTF-8” och inte “utf-8”), så det är säkrast att i de flesta fall välja canonical namn för en särskild locale. För att avgöra det canonical namnet, kör följande kommando, där <locale name> är output från **locale -a** för din föredragna locale “en_GB.iso88591” i detta exempel.

```
LC_ALL=<locale name> locale charmap
```

För locale “en_GB.iso88591” kommer kommandot ovan att skriva:

```
ISO-8859-1
```

Detta resulterar i en slutgiltig locale inställning av “en_GB.ISO-8859-1”. Det är viktigt att den locale som återfinns med hjälp av heuristiken ovan, testas innan det att den adderas till Bashs uppstarts filer:

```
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
```

Ovan kommandon bör skriva språkets namn, karaktärs encoding som används av vald locale, den lokala valutan, samt vilket prefix som används för att ringa till landet. Ifall något av kommandona ovan misslyckas med ett meddelande som liknar det nedan, innebär detta att din locale antingen inte installerades i kapitel 6 eller inte stödjs av standard installationen av Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Ifall detta sker, borde du antingen installera den önskade localen med hjälp av kommandot **localedef** eller överväga att välja en annan locale. Här efterföljande instruktioner utgår ifrån att ingan sådana error meddelanden mottogs.

Vissa paket som finns bortom LFS kan också sakna stöd för dina valda locales. Ett exempel är biblioteket för X (del av X Window System), vilket skickar följande output ifall localen inte exakt matchar en av character mapsen i dess interna filer.

```
Warning: locale not supported by Xlib, locale set to C
```

I flertalet fall förväntar sig Xlib att character mapen kommer listas i stora bokstäver med canonical bindesträck. T.ex “ISO-8859-1” istället för “iso88591”. Det är också möjligt att hitta en lämplig specifikation genom att radera charmap delen av specifikationen för en locale. Detta kan kollas genom att köra kommandot **locale charmap** i båda locales. Exempelvis skulle man behöva ändra “de_DE.ISO-8859-15@euro” till “de_DE@euro” för att lyckas få denna locale igenkänd av Xlib.

Andra paket kan också funka inkorrekt (men behöver inte nödvändigtvis visa några error meddelanden) ifall namnet på en locale inte motsvarar förväntningarna. i dessa fall kan man skaffa sig relevant information genom att efterforska hur andra Linux distros hanterar den problematiska localen.

När de korrekta locale inställningarna väl är bestämda, skapa filen /etc/profile:

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

export LANG=<ll>_<CC>.<charmap><@modifiers>

# End /etc/profile
EOF
```

“C” (standard) och “en_US” (den rekommenderade för användare i Förenta Staterna) locales är olika. “C” använder karaktärs-setet US-ASCII 7-bit, och behandlar bytes med den översta biten specifierad som invalida karaktärer. Det är därför som kommandot **ls** ersätter dem med frågetecken i den localen. Ett försök att skicka mail med sådana karaktärer från Mutt eller Pine resulterar i att non-RFC-conforming meddelanden skickas (charset i det utgående mailet indikeras som “unknown 8-bit”). Så du kan använda endast locale “C” ifall du är säkert på att du aldrig kommer behöva 8-bits karaktärer.

UTF-8 baserade locales stödjs inte så bra av vissa program. Arbetet pågår för tillfället för att dokumentera och, ifall nödvändigt, fixa sådana problem. Referera till www.linuxfromscratch.org/blfs/view/9.1/introduction/locale-issues.html.

7.8. Skapa filen /etc/inputrc

Filen inputrc är konfigurationsfilen för Readline biblioteket, vilket tillhandahåller redigerings funktionalitet medan det att användaren skriver in en rad via terminalen. Det fungerar genom att översätta tangentbords input till specifika handlingar. Readline används av Bash samt de flesta andra skal, och även av flertalet applikationer.

De flesta personer har inget behov av användar-specifik funktionalitet, så kommandot nedan skapar en global /etc/inputrc fil som används av alla som loggar in. Om du senare bestämmer dig för att du vill kunna tillhandahålla användar-specifik funktionalitet, kan du skapa en .inputrc fil, med de modifierade mappingsen, i användarens home mapp.

För mer information angående hur filen inputrc kan redigeras, refera till **info bash** och sektionen *Readline Init File*. **info readline** tillhandahåller också användbar information.

Nedan återfinns en generisk och global inputrc fil, tillsammans med kommentarer som förklarar vad de olika alternativen gör. Notera att kommentarer inte kan skrivas på samma rad som kommandon. Skapa filen med följande kommando:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions
"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

7.9. Skapa filen /etc/shells

Filen shells innehåller en lista med login skal som existerar på systemet. Applikationer använder denna fil för att avgöra ifall ett skal är giltigt. För varje skal skall en enskild rad finnas tillgänglig, vilken består av skalets sökväg relativ till mappstrukturens root (/).

Denna fil konsulteras exempelvis av **chsh** för att avgöra om en opriviligerad användare har tillåtelse att ändra login skalet för sitt eget konto. Ifall kommandots namn inte listas, tillåts inte användaren att ändra.

It is a requirement for applications such as GDM which does not populate the face browser if it can't find /etc/shells, or FTP daemons which traditionally disallow access to users with shells not included in this file.

```
cat > /etc/shells << "EOF"
# Begin /etc/shells

/bin/sh
/bin/bash

# End /etc/shells
EOF
```

Kapitel 8. Att göra LFS-Systemet uppstartsbart

8.1. Introduktion

Det är nu dags att göra systemet uppstartsbart. Detta kapitel diskuterar skapandet av en fstab fil, byggandet av en kernel för det nya LFS systemet, samt installation av GRUB boot loadern så att LFS kan väljas vid uppstart av datorn.

8.2. Skapa filen /etc/fstab

Filen /etc/fstab används av vissa program för att avgöra var filsystem ska monteras som standard, i vilken ordning, samt vilka som måste integritetskollas före det att de monteras. Skapa ett nytt filsystem table:

```
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

# End /etc/fstab
EOF
```

Ersätt *<xxx>*, *<yyy>*, och *<fff>*Med värdena som är lämpliga för systemet, exempelvis `sda2`, `sda5`, och `ext4`. För detaljerad information vad gäller de sex fälten i denna fil, se **man 5 fstab**.

Filsystem med ursprung i MS-DOS eller Windows (t.ex vfat, ntfs, smbfs, cifs, iso9660, udf) kräver ett särskilt alternativ, utf-8, för att icke-ASCII karaktärer i filnamn ska tolkas korrekt. För icke-UTF-8 locales, skall värdet för iocharset sättas till samma som karaktärs-setet för korrekt locale, justerat på ett sådant vis att kerneln kan förstå det. Detta fungerar ifall den relevanta karaktärs-sets definitionen (hittad under File systems → Native Language Support vid konfiguration av kerneln) har kompilerats in i kerneln eller byggts som en modul. Ifall character-setet för localen är UTF-8 däremot, skulle alternativet iocharset=utf8 göra filsystemet känsligt för stora/små bokstäver. För att fixa detta för UTF-8 locales, används alternativet utf8 istället för iocharset=utf8. Alternativet “codepage” behövs också för vfat och smbfs filsystem. Det ska sättas till det codepage nummer som används för MS-DOS i ditt land.
För att exempelvis montera USB minnen, behöver en ru_RU.KO18-R användare ha följande text inskriven i alternativs sektionen för montering i /etc/fstab:

```
noauto,user,quiet,showexec,codepage=866,iocharset=koi8r
```

Det motsvarande alternativs fragmentet för ru_RU.UTF-8 användare är:

```
noauto,user,quiet,showexec,codepage=866,utf8
```

Notera att användandet av iocharset är standard för iso8859-1 (som gör att filsystemet inte är känsligt för stora/små bokstäver, och att utf-8 alternativet specifierar för kerneln att konvertera filnamnen med hjälp av UTF-8 så att de kan tolkas med hjälp av UTF-8 localen.

Det är också möjligt att specifiera standard codepage och iocharset värden för vissa filsystem vid kernel konfigurationen. De relevanta parametrarna kallas “Default NLS Option” (CONFIG_NLS_DEFAULT), “Default Remote NLS Option” (CONFIG_SMB_NLS_DEFAULT), “Default codepage for FAT” (CONFIG_FAT_DEFAULT_CODEPAGE), och “Default iocharset for FAT” (CONFIG_FAT_DEFAULT_IOCHARSET). Det går inte att specifiera dessa inställningar för filsystemet ntfs vid kompilering av kerneln.

Det är möjligt att för vissa typer av hårddiskar göra ext3 filsystemet pålitligt i händelse av problem med strömförsörjning. Lägg till monterings alternativet barrier=1 till den korrekta entryn i /etc/fstab. För att kolla ifall disken stödjer detta alternativ, kör hdparm på den berörda disken. Till exempel:

```
hdparm -I /dev/sda | grep NCQ
```

Ifall detta returnerar icke-tom output, stödjs alternativet.

Notera att Logical Volume Management (LVM) baserade partitioner inte kan använda alternativet barrier.

8.3. Linux-5.5.3

Paketet Linux innehåller Linux kerneln.

Förväntad byggtid: 4.4 - 66.0 SBU (vanligtvis omkring 6 SBU)
Nödvändigt disk-utrymme: 960 - 4250 MB (vanligtvis omkring 1100 MB)

8.3.1. Installation av kerneln

Att bygga kerneln inkluderar ett antal steg – konfiguration, kompilation, och installation. Läs filen README i source trädet för kerneln, för alternativa metoder vad gäller konfiguration av kerneln.

Förbered kompilering:

```
make mrproper
```

Detta säkerställer att kernelträdet är helt rent. Kernel teamet rekommenderar att detta kommando körs inför varje enstaka kernel kompilering. Förvänta dig inte att source trädet är rent efter att ha packats upp. Säkerställ det själv.

Konfigurera kerneln via ett meny-drivet gränssnitt. För generell information vad gäller konfigurationen, referera till http://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt. BLFS innehåller en del information vad Gäller särskilda konfigurations-krav för paket utanför LFS: http://www.linuxfromscratch.org/blfs/view/9.1/longindex.html#kernel-config-index. Ytterligare information vad gäller konfigurering samt kompilering av kerneln kan hittas via http://www.kroah.com/lkn/

Kommentar

En bra början när det kommer till att specifiera kernelns konfiguration är att köra **make defconfig**. Detta kommer sätta den grundläggande konfigurationen till ett bra tillstånd vilket tar med det nuvarande systemets arkitektur i beräkningarna för relevanta inställningar.

Aktivera/avaktivera/specifiera följande funktionalitet, eller så kanske systemet inte kommer funka eller boota.

```
Device Drivers  --->
  Generic Driver Options  --->
    [ ] Support for uevent helper [CONFIG_UEVENT_HELPER]
    [*] Maintain a devtmpfs filesystem to mount at /dev [CONFIG_DEVTMPFS]
Kernel hacking  --->
        Choose kernel unwinder (Frame pointer unwinder)  ---> [CONFIG_
UNWINDER_FRAME_POINTER]
```

Det finns flertalet övriga inställningar som kan vara önskvärda beroende på systemkraven. För en lista över alternativ som krävs för BLFS, referera till BLFS Index of Kernel Settings (http://www.linuxfromscratch.org/blfs/view/9.1/longindex.html#kernel-config-index).

Kommentar

Ifall din värds hårdvara använder UEFI, då borde “make defconfig” ovan automatiskt lägga till diverse EFI-relaterade kernel alternativ.

För att tillåta din LFS kernel att bootas från inut din värds UEFI bott miljö, måste din kernel ha alternativet nedan markerat:

```
Processor type and features  --->
   [*]   EFI stub support  [CONFIG_EFI_STUB]
```

En mer omfattnade beskrivning vad gäller hanteringen av UEFI miljöer från inuti LFS, täcks av lfs-uefi.txt på: http://www.linuxfromscratch.org/hints/downloads/files/lfs-uefi.txt.

Orsakerna till konfigurations-alternativen som väljs ovan:

`Support for uevent helper`
Att ha detta alternativt specifierat kan inverka på enhetshanteringen då Udev/Eudev används.

`Maintain a devtmpfs`
Detta kommer skapa automatiserade enhets-noder vilka populeras av kerneln, även ifall Udev inte körs. Udev körs sedan ovanpå detta, hanterande rättigheter och adderande symlänkar. Detta alternativ krävs för alla användare av Udev/Eudev.

```
make menuconfig
```

Vad alternativen betyder:

`LANG=<host_LANG_value> LC_ALL=`
Detta etablerar locale inställningen till den som används på värden. Detta kan vara nödvändigt för ett korrekt menuconfig ncurses interface line drawing via en UTF-8 linux text-konsol.
Om använd, säkerställ att <host_LANG_value> ersätts med din värdet i din värds $LANG variabel. Alternativt kan du använda värdet i variabeln $LC_ALL eller $LC_CTYPE.

Det kan även var så att **make oldconfig** ibland är lämpligar i vissa situationer. Läs `README` för mer information.

Om du önskar, går det hoppa över konfigurationen av kerneln genom att kopiera kernels konfigurations fil, .config, från värdsystemet (förutsatt att det finns tillgängligt) till mappen linux-5.5.3. Detta alternativ rekommenderas dock inte. Det är oftast bättre att utforska alla konfigurations-menyer och skapa konfigurationen från grunden.

Kompilera kernel imagen och moduler:

```
make
```

Vid användning av kernel moduler, kan konfiguration i /etc/modprobe.d krävas. Information som behandlar konfiguration av moduler och kernel återfinns i sektion 7.3 och i kernel dokumentationen i mappen linux-5.5.3/Documentation. Även modprobe(5) kan vara av intresse.

Bortsett från ifall stöd för moduler inaktiverats, installera modulerna med:

```
make modules_install
```

Efter att kompileringen är fullbordad, krävs ytterligare steg för att slutföra installationen. Vissa filer behöver kopieras till mappen /boot.

Var uppmärksam

Ifall värdsystemet har en separat /boot partition, ska de kopierade filerna hamna där. Det enklaste sättet att få detta att ske är genom att binda /boot på värden (utanför chroot) till /mnt/lfs/boot. Kör, som användare root på värdsystemet:

```
mount --bind /boot /mnt/lfs/boot
```

Sökvägen till kernel imagen kan variera beroende på vilken plattform som används. Filnamnet nedan kan ändras till ett du tycker passar, men ska börja med vmlinuz för att vara kompatibelt med den automatiska konfigurationen av boot-processen som beskrivs i nästa sektion. Följande kommando förväntar sig en x86 arkitektur:

```
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-5.5.3-lfs-9.1
```

System.map är en symbol-fil för kerneln. Den mappar funktioners entry points för varje funktion i kernelns API, och även adresserna för den aktiva kernelns data strukturer. Den används som en resurs då man undersöker problem med kerneln. Kör följande kommando för att installera map-filen:

```
cp -iv System.map /boot/System.map-5.5.3
```

Kernelns konfigurations-fil .config som produceras av steget **make config** ovan, innehåller alla konfigurations-alternativ för den kernel som just kompilerades. Det är en bra idé att behålla denna fil för att i framtiden kunna referera till:

```
cp -iv .config /boot/config-5.5.3
```

Installera dokumentationen för Linux kerneln:

```
install -d /usr/share/doc/linux-5.5.3
cp -r Documentation/* /usr/share/doc/linux-5.5.3
```

Det är viktigt att notera att filerna i kernel source-mappen inte ägs av root. Då ett paket extraheras som användaren root (som vi gjorde inuti chroot), har filerna användare och grupp ID identiskt med vad de hade då de paketerades, även om detta skedde på en annan dator. Detta innebär vanligtvis inte ett problem för andra paket som ska installeras, då source-trädet raderas efter installationen. Linux-trädet behålls dock ofta under en längre tid. På grund av detta finns det en chans att det användar-ID som användes för att packa paketet, kommer tillskrivas en användare på systemet. Denna person skulle då få tillgång till kernelns source.

Kommentar

i många fall kommer konfigurationen av kerneln att behövs uppdateras för paket som kommer installeras senare i BLFS. Till skillnad från andra paket, är det inte nödvändigt att radera kernelns source-träd efter att den nybyggda kerneln blivit installerad.

Ifall kernelns source-träd ska bibehålls i system kör då **chown -R 0:0** på `linux-5.5.3` mappen för att säkerställa att alla filer ägs av användaren root.

Varning

En del av kernel dokumentationen rekommenderar skapandet av en symlänk från /usr/src/linux som pekar till kernelns source mapp. Detta är specifikt för kernels innan 2.6 serien och ***skall inte*** skapas på ett LFS-system då det kan leda till problem för paket som du kanske önskar bygga framöver.

Varning

Headersen i systemets include-mapp (/usr/include) ska alltid vara de vilka glibc kompilerades gentemot. Alltså de sanitiserade headersen installerade i sektion 6.7. De ska aldrig ersättas av vare sig råa kernel headers eller några andra sanitiserade headers.

8.3.2. Konfigurera Laddning-ordningen för Linux moduler

För det mesta laddas Linux moduler automatiskt. Ibland behöver dock specifika direktiv utfärdas. Programmet vilket laddar moduler, **modprobe** eller **insmod**, använder för detta syfte /etc/modprobe.d/usb.conf. Denna fil behöver skapas så att USB drivrutiner (ehci_hcd, ohci_hcd and uhci_hcd, och om de byggts som moduler) kommer laddas i korrekt ordning. Ifall de laddas i fel ordning kommer varningar rapporteras vid uppstart.

Skapa filen `/etc/modprobe.d/usb.conf` genom att köra följande:

```
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
```

8.3.3. Innehåll Linux

Installerade filer: config-5.5.3, vmlinuz-5.5.3-lfs-9.1, och System.map-5.5.3
Installerade mappar: /lib/modules, /usr/share/doc/linux-5.5.3

Korta beskrivningar

`config-5.5.3` Innehåller alla konfigurations-alternativ för kerneln

`vmlinuz-5.5.3-lfs-9.1` Linux-systemets motor. Då datorn sätts på är kerneln den första delen av operativ-systemet som laddas. Den upptäcker och initialiserar alla datorns hårdvarukomponenter och tillgängliggör sedan dessa komponenter som ett filträd för mjukvaran, och omvandlar en enskild CPU till en multitasking maskin kapabel att köra flertalet program samtidigt (eller rättare sagt, vad som verkar som samtidigt).

`System.map-5.5.3` En lista med adresser och symboler; den kartlägger entry points och adresser för alla funktioner och data-strukturer i kerneln.

8.4. Att använda GRUB för att definiera uppstarts-processen

8.4.1. Introduktion

Varning

Att konfigurera GRUB på ett inkorrekt vis kan leda till att ditt system inte fungerar ifall det inte finns en sekundär metod för att boota (t.ex en CD-ROM). Denna sektion krävs inte för att det ska gå boota ditt system. Du kanske bara vill modifiera din nuvarande Boot-loader (Grub-Legacy, GRUB2, eller LILO).

Säkerställ att det finns en “säkerhet” disk med vilken det går boota datorn, ifall något går fel och datorn inte kan bootas. Ifall du inte redan har en boot-enhet kan du skapa en. För att proceduren nedan ska fungera, behöver du använda BLFS och installera xorriso från paketet libisoburn.

```
cd /tmp
grub-mkrescue --output=grub-img.iso
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

Kommentar

För att boota LFS på system vilka har UEFI aktiverat, måste kerneln har byggts med funktionalitet för CONFIG_EFI_STUB, beskrivet i den tidigare sektionen. LFS kan däremot bootas med hjälp av GRUB2 utan ett sådant tillägg. För att göra detta, måste UEFI Mode och Secure Boot funktionalitet i värdens system BIOSs stängas av. För detaljer, läs lfs-uefi.txt via http://www.linuxfromscratch.org/hints/downloads/files/lfs-uefi.txt

8.4.2. GRUBs konventioner för namngivning

GRUB använder sin egen namngivnings struktur för diskar och partitioner i formen av (hdn,m). Där “n” är hårddisks-numret och “m” är partitionsnumret. Numren för hårddisk börjar från 0, men partitions-nummer börjar från 1 för vanliga partitioner och 5 för förlängda. Notera att detta är annorlunda från tidigare versioner där båda nummer startade från 0. Exempelvis partition sda1 är (hd0,1) för GRUB och sdb3 (hd1,3). i kontrast till Linux betraktar GRUB inte CD-ROMs som om de vore hårddiskar. Ifall en CD används på hdb och en sekundär hårddisk på hdc, vore den andra hårdisken (hd1).

8.4.3. Etablera konfigurationen

GRUB fungerar genom att skriva data till den första fysiska remsan på hårddisken. Detta utrymme är inte del av något filsystem. Programmen där skaffar sig tillgång till GRUB moduler i boot partitionen. Standard plats är /boot/grub.

Platsen för boot partitionen är ett användarval som påverkar hela konfigurationen. En rekommendation är att ha en separat liten partition (ca 100mb) endast för boot information. På detta sätt kan varje system, vare sig LFS eller kommersiellt, ha tillgång till samma boot-filer och tillgång till dessa kan ske från alla bootade system. Ifall du önskar göra detta, måste du montera den separata partitionen, flytta alla filer i den nuvarande /boot mappen (alltså Linux kerneln du byggde i den föregående sektionen) till den nya partitionen. Du kommer sedan behöva avmontera partitionen och återmontera den som /boot. Ifall du gör detta, se till att du även uppdatera /etc/fstab.

Att använda den nuvarande lfs partitionen fungerar, men konfigurationen för multipla system blir då mer komplicerad.

Genom att använda ovan information, avgör den lämpliga designationen för root partitionen (eller boot partitionen, ifall en separat sådan ska användas). i följande exempel, utgår vi från att root (eller separat boot) partitionen är sda2.

Installera GRUB filerna i /boot/grub och sätt upp boot spåret:

Varning

Följande kommando kommer att skriva över nuvarande boot loader. Kör inte kommandot ifall detta inte är önskvärt, t.ex ifall du använder en tredjeparts boot manager för att hantera Master Boot Record (MBR).

```
grub-install /dev/sda
```

Kommentar

Ifall systemet har bootats med UEFI, kommer **grub-install** försöka installera filer för *x86_64-efi* målet, men dessa filer har inte installerats i kapitel 6. Ifall detta är fallet, lägg till --target i386-pc till kommandot ovan.

8.4.4. Skap GRUBs konfigurations-fil

Generera `/boot/grub/grub.cfg`:

```
cat > /boot/grub/grub.cfg << "EOF"
# Begin /boot/grub/grub.cfg
set default=0
set timeout=5

insmod ext2
set root=(hd0,2)

menuentry "GNU/Linux, Linux 5.5.3-lfs-9.1" {
        linux   /boot/vmlinuz-5.5.3-lfs-9.1 root=/dev/sda2 ro
}
EOF
```

Kommentar

Från GRUBs perspektiv, är kernel-filerna relativa till partitionen som används. Ifall du använde en separat /boot partition, ta bort /boot från raden “linux” ovan. Du kommer också behöva ändra raden “set root” så att den pekar till boot partitionen.

GRUB är ett extremt kraftfullt program som tillhandahåller en väldig mängd alternativ för att boota ett stort spektrum enheter, operativsystem, och partitionstyper. Det finns även många alternativ för grafiska uppstarts skärmar, ljud, mus input, etc. Detaljerna vad gäller dessa alternativ befinner sig bortom syftet med denna introduktion.

Var uppmärksam

Det finns ett kommando, grub-mkconfig, som kan skriva en konfigurations-fil automatiskt. Det använder en samling skript i /etc/grub.d/ och kan förstöra alla anpassningar som du gör. Dessa skript är huvudsakligen designade för non-source distros och rekommenderas inte för LFS. Ifall du installerar en kommersiell distro, finns det däremot en god chans att detta program kommer köras. Se till att backa upp din grub.cfg fil.

Kapitel 9. Avslutning

9.1. Avslutning

Bra jobbat! Ditt nya LFS-system är nu installerat. Vi önskar dig lycka till med ditt specialbyggda Linux-system.

Det kan vara en god idé att skapa en /etc/lfs-release fil. Genom att ha denna fil tillgänglig, blir det väldigt enkelt för dig (och för oss om du behöver fråga om hjälp någon gång) att ta reda på vilken LFS-version som är installerad. Skapa denna fil genom att köra:

```
echo 9.1 > /etc/lfs-release
```

Två filer som beskriver det installerade systemet kan komma att användas av paket som kommer att installeras på systemet, antingen i binär form eller genom att bygga dem.

Den första visar ditt nya systems status enligt Linux Standard Base(LSB). För att skapa denna fil, kör:

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="9.1"
DISTRIB_CODENAME="<your name here>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

Den andra filen innehåller med eller mindre samma information, och används av systemd samt diverse grafiska skrivbordsmiljöer. För att skapa denna fil, kör:

```
cat > /etc/os-release << "EOF"
NAME="Linux From Scratch"
VERSION="9.1"
ID=lfs
PRETTY_NAME="Linux From Scratch 9.1"
VERSION_CODENAME="<your name here>"
EOF
```

Se till att lägga till information för fälten 'DISTRIB_CODENAME' och 'VERSION_CODENAME' för att göra systemet till ett unikt sådant.

9.2. Bli inräknad

Då du nu har slutfört systemet, vill du bli inräknad som en LFS-användare? Ta dig då till http://www.linuxfromscratch.org/cgi-bin/lfscounter.php och registrera dig som en LFS-användare genom att skriva in ditt namn och den första version av LFS som du har använt.

Det är nu dags att starta om datorn in till LFS.

9.3. Omstart av systemet

Då all mjukvara nu har installerats, är det dags att starta om datorn. Du bör däremot vara medveten om ett antal saker. Systemet vilket du har skapat med hjälp av denna bok är relativt minimalistiskt, och kommer antagligen inte innehålla den funktionalitet vilken du faktiskt behöver. Genom att installera ett antal paket från boken BLFS medan du fortfarande är i den nuvarande chroot miljön, kan du skaffa dig en mycket lämpligare system innan du bootar in i LFS.
Här nedan återfinns ett antal rekommendationer vad gäller ytterligare paket:

- En text hanterare såsom Lynx kommer tillåta dig att enkelt visa boken BLFS i en virtuell terminal, samtidigt som du bygger paket i en annan.
- Paketet GPM kommer tillåta dig att klippa/klistra emellan olika virtuella terminaler.
- Ifall du befinner dig i en situation där statisk IP-konfiguration inte lever upp till nätverks-kraven, installera isåfall ett paket likt dhcpcd eller dhcp.
- Att installera sudo kan vara av nytta för att bland annat bygga och installera paket som en icke-root användare.
- Ifall du vill kunna ha GUI-tillgång till ditt system från ett annat system, installera då openssh.
- För att göra det enklare att hämta filer från internet, installera wget.
- Ifall en eller flera av dina diskar har ett GUID partitions table (GPT), är antingen *gptfdisk* eller *parted* användbart.
- Slutligen, så är en genomgång och dubellkoll av konfigurationsfilerna även lämplig:
  - /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

Med dessa sista ord, är det nu dags att en första gång boota in i det nya LFS-systemet. Lämna först chroot miljön:

```
logout
```

Avmontera sedan de virtuella filsystemen:

```
umount -v $LFS/dev/pts
umount -v $LFS/dev
umount -v $LFS/run
umount -v $LFS/proc
umount -v $LFS/sys
```

Avmontera LFS-filsystem självt:

```
umount -v $LFS
```

Ifall multipla partitioner skapades, avmontera de övriga partitionerna innan den huvudsakliga sådana:

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

Starta nu om systemet:

```
shutdown -r now
```

Förutsatt att GRUB boot loadern konfigurerades som specifierat, är menyn satt att automatiskt boota LFS 9.1.

När omstarten är färdig, kommer LFS-systemet vara redo för anvädning och ytterligare mjukvara kan då installeras.

9.4. Och nu?

Tack för att du läste denna bok. Vi hoppas att du har funnit den användbar och har lärt dig en hel del om processen vad gäller att skapa ett Linux system.

Nu då LFS-systemet är installerat kanske du undera “Och nu?”. För att besvara den frågan har vi sammanställt en lista med resurser för dig:

- Underhåll

  Buggar och säkerhetsnotiser rapporteras frekvent för all mjukvara. Eftersom ett LFS-system kompileras från källorna, är det upp till dig att vara medveten om sådana rapporter. Det finns flertalet resurser online som spårar sådana rapporter, en del av dessa återfinns i listan nedan:

  - *CERT* (Computer Emergency Response Team)

    CERT har en mail-lista som publicerar säkerhetsvarningar för diverse operativsystem och applikationer. Information om prenumeration finns tillgänglig här: http://www.us-cert.gov/cas/signup.html

  - Bugtraq

    Bugtraq är också en mail-lista för säkerhetsproblem, för en stor mängd mjukvara. Information om prenumeration finns tillgänglig här: http://www.securityfocus.com/archive

- Beyond Linux From Scratch

  Boken Beyond Linux From Scratch täcker installations procedurer för en stor mängd mjukvara bortom LFS-boken. Projektet BLFS kan hittas här: http://www.linuxfromscratch.org/blfs/

- LFS Hints

  LFS Hints är en samling dokument inskickade av LFS volontärer. Dessa tips kan återfinns här: http://www.linuxfromscratch.org/hints/list.html.

- Mailing lists

  Det finns flertalet LFS mail-listor vilka du kan prenumera på ifall du behöver hjälp, vill hålla dig uppdatera vad gäller de senaste utvecklingarna, vill bidra till projektet, eller annat. Se kapitel 1 – mail-listor för mer information.

- The Linux Documentation Project

  Målet med The Linux Documentation Project (TLDP) är att samarbeta vad gäller problem med Linux-dokumentation. TDLP innehåller en stor mängd HOWTOs, guider, och man sidor. Projektet återfinns här: http://www.tldp.org/.

Del IV. Appendix

Appendix A. Acronyms and Terms

ABI Application Binary Interface
ALFS Automated Linux From Scratch
API Application Programming Interface
ASCII American Standard Code for Information Interchange
BIOS Basic Input/Output System
BLFS Beyond Linux From Scratch
BSD Berkeley Software Distribution
chroot change root
CMOS Complementary Metal Oxide Semiconductor
COS Class Of Service
CPU Central Processing Unit
CRC Cyclic Redundancy Check
CVS Concurrent Versions System
DHCP Dynamic Host Configuration Protocol
DNS Domain Name Service
EGA Enhanced Graphics Adapter
ELF Executable and Linkable Format
EOF End of File
EQN equation
ext2 second extended file system
ext3 third extended file system
ext4 fourth extended file system
FAQ Frequently Asked Questions
FHS Filesystem Hierarchy Standard
FIFO First-In, First Out
FQDN Fully Qualified Domain Name
FTP File Transfer Protocol
GB Gigabytes
GCC GNU Compiler Collection
GID Group Identifier
GMT Greenwich Mean Time
HTML Hypertext Markup Language
IDE Integrated Drive Electronics
IEEE Institute of Electrical and Electronic Engineers

IO Input/Output
IP Internet Protocol
IPC Inter-Process Communication
IRC Internet Relay Chat
ISO International Organization for Standardization
ISP Internet Service Provider
KB Kilobytes
LED Light Emitting Diode
LFS Linux From Scratch
LSB Linux Standard Base
MB Megabytes
MBR Master Boot Record
MD5 Message Digest 5
NIC Network Interface Card
NLS Native Language Support
NNTP Network News Transport Protocol
NPTL Native POSIX Threading Library
OSS Open Sound System
PCH Pre-Compiled Headers
PCRE Perl Compatible Regular Expression
PID Process Identifier
PTY pseudo terminal
QOS Quality Of Service
RAM Random Access Memory
RPC Remote Procedure Call
RTC Real Time Clock
SBU Standard Build Unit
SCO The Santa Cruz Operation
SHA1 Secure-Hash Algorithm 1
TLDP The Linux Documentation Project
TFTP Trivial File Transfer Protocol
TLS Thread-Local Storage
UID User Identifier
umask user file-creation mask
USB Universal Serial Bus
UTC Coordinated Universal Time

UUID Universally Unique Identifier

VC Virtual Console

VGA Video Graphics Array

VT Virtual Terminal

Appendix B. Acknowledgments

We would like to thank the following people and organizations for their contributions to the Linux From Scratch Project.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – LFS Creator
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – LFS Managing Editor
- *Jim Gifford* <jim@linuxfromscratch.org> – CLFS Project Co-Leader
- *Pierre Labastie* <pierre@linuxfromscratch.org> – BLFS Editor and ALFS Lead
- *DJ Lucas* <dj@linuxfromscratch.org> – LFS and BLFS Editor
- *Ken Moffat* <ken@linuxfromscratch.org> – BLFS Editor
- Countless other people on the various LFS and BLFS mailing lists who helped make this book possible by giving their suggestions, testing the book, and submitting bug reports, instructions, and their experiences with installing various packages.

Translators

- *Manuel Canales Esparcia* <macana@macana-es.com> – Spanish LFS translation project
- *Johan Lenglet* <johan@linuxfromscratch.org> – French LFS translation project until 2008
- *Jean-Philippe Mengual* <jmengual@linuxfromscratch.org> – French LFS translation project 2008-2016
- *Julien Lepiller* <jlepiller@linuxfromscratch.org> – French LFS translation project 2017-present
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Portuguese LFS translation project
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – German LFS translation project
- *Anton Maisak* <info@linuxfromscratch.org.ru> – Russian LFS translation project
- *Elena Shevcova* <helen@linuxfromscratch.org.ru> – Russian LFS translation project

Mirror Maintainers

North American Mirrors

- *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

South American Mirrors

- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – lfsmirror.lfs-es.info mirror
- *Luis Falcon* <Luis Falcon> – torredehanoi.org mirror

European Mirrors

- *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

Asian Mirrors

- *Satit Phermsawang* <satit@wbac.ac.th> – lfs.phayoune.org mirror
- *Shizunet Co.,Ltd.* <info@shizu-net.jp> – lfs.mirror.shizu-net.jp mirror
- *Init World* <http://www.initworld.com/> – lfs.initworld.com mirror

Australian Mirrors

- *Jason Andrade* <jason@dstc.edu.au> – au.linuxfromscratch.org mirror

Former Project Team Members

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – LFS Book Editor
- Archaic <archaic@linuxfromscratch.org> – LFS Technical Writer/Editor, HLFS Project Leader, BLFS Editor, Hints and Patches Project Maintainer
- *Matthew Burgess* <matthew@linuxfromscratch.org> – LFS Project Leader, LFS Technical Writer/Editor
- *Nathan Coulson* <nathan@linuxfromscratch.org> – LFS-Bootscripts Maintainer
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Website Developer, FAQ Maintainer
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – LFS/BLFS/HLFS XML and XSL Maintainer
- Alex Groenewoud – LFS Technical Writer
- Marc Heerdink
- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – LFS Technical Writer, LFS LiveCD Maintainer
- *Bryan Kadzban* <bryan@linuxfromscratch.org> – LFS Technical Writer
- Mark Hymers

- Seth W. Klein – FAQ maintainer
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Wiki Maintainer
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Website Backend-Scripts Maintainer
- *Randy McMurchy* <randy@linuxfromscratch.org> – BLFS Project Leader, LFS Editor
- *Dan Nicholson* <dnicholson@linuxfromscratch.org> – LFS and BLFS Editor
- *Alexander E. Patrakov* <alexander@linuxfromscratch.org> – LFS Technical Writer, LFS Internationalization Editor, LFS Live CD Maintainer
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – LFS NNTP Gateway Maintainer
- *Douglas R. Reno* <renodr@linuxfromscratch.org> – Systemd Editor
- *Ryan Oliver* <ryan@linuxfromscratch.org> – CLFS Project Co-Leader
- *Greg Schafer* <gschafer@zip.com.au> – LFS Technical Writer and Architect of the Next Generation 64-bit-enabling Build Method
- Jesse Tie-Ten-Quee – LFS Technical Writer
- *James Robertson* <jwrober@linuxfromscratch.org> – Bugzilla Maintainer
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – BLFS Book Editor, Hints and Patches Project Leader
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – LFS Technical Writer, Bugzilla Maintainer, LFS-Bootscripts Maintainer
- *Zack Winkles* <zwinkles@gmail.com> – LFS Technical Writer

Appendix C. Dependencies

Every package built in LFS relies on one or more other packages in order to build and install properly. Some packages even participate in circular dependencies, that is, the first package depends on the second which in turn depends on the first. Because of these dependencies, the order in which packages are built in LFS is very important. The purpose of this page is to document the dependencies of each package built in LFS.

For each package we build, we have listed three, and sometimes four, types of dependencies. The first lists what other packages need to be available in order to compile and install the package in question. The second lists what packages, in addition to those on the first list, need to be available in order to run the test suites. The third list of dependencies are packages that require this package to be built and installed in its final location before they are built and installed. In most cases, this is because these packages will hard code paths to binaries within their scripts. If not built in a certain order, this could result in paths of /tools/bin/[binary] being placed inside scripts installed to the final system. This is obviously not desirable.

The last list of dependencies are optional packages that are not addressed in LFS, but could be useful to the user. These packages may have additional mandatory or optional dependencies of their own. For these dependencies, the recommended practice is to install them after completion of the LFS book and then go back and rebuild the LFS package. In several cases, re-installation is addressed in BLFS.

Acl

Installation depends on: Attr, Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed, and Texinfo
Test suite depends on: Automake, Diffutils, Findutils, and Libtool
Must be installed before: Coreutils, Sed, Tar, and Vim
Optional dependencies: None

Attr

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed, and Texinfo
Test suite depends on: Automake, Diffutils, Findutils, and Libtool
Must be installed before: Acl and Libcap
Optional dependencies: None

Autoconf

Installation depends on: Bash, Coreutils, Grep, M4, Make, Perl, Sed, and Texinfo
Test suite depends on: Automake, Diffutils, Findutils, GCC, and Libtool
Must be installed before: Automake
Optional dependencies: Emacs

Automake

Installation depends on: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed, and Texinfo
Test suite depends on: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, and Tar
Must be installed before: None
Optional dependencies: None

Bash

Installation depends on: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed, and Texinfo
Test suite depends on: Shadow
Must be installed before: None
Optional dependencies: Xorg

Bc

Installation depends on: Bash, Binutils, Bison, Coreutils, GCC, Glibc, Grep, Make, Perl, and Readline
Test suite depends on: Gawk
Must be installed before: Linux Kernel
Optional dependencies: None

Binutils

Installation depends on: Bash, Binutils, Coreutils, Diffutils, File, Gawk, GCC, Glibc, Grep, Make, Perl, Sed, Texinfo and Zlib
Test suite depends on: DejaGNU and Expect
Must be installed before: None
Optional dependencies: None

Bison

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl, and Sed
Test suite depends on: Diffutils, Findutils, and Flex
Must be installed before: Kbd and Tar
Optional dependencies: Doxygen (test suite)

Bzip2

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, and Patch
Test suite depends on: None
Must be installed before: File
Optional dependencies: None

Check

Installation depends on: GCC, Grep, Make, Sed, and Texinfo
Test suite depends on: None
Must be installed before: None
Optional dependencies: None

Coreutils

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Libcap, Make, Patch, Perl, Sed, and Texinfo
Test suite depends on: Diffutils, E2fsprogs, Findutils, Shadow, and Util-linux
Must be installed before: Bash, Diffutils, Eudev, Findutils, and Man-DB
Optional dependencies: Perl Expect and IO:Tty modules (for test suite)

DejaGNU

Installation depends on: Bash, Coreutils, Diffutils, GCC, Grep, Make, and Sed
Test suite depends on: None
Must be installed before: None
Optional dependencies: None

Diffutils

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: Perl
Must be installed before: None
Optional dependencies: None

E2fsprogs

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Sed, Texinfo, and Util-linux
Test suite depends on: Procps-ng and Psmisc
Must be installed before: None
Optional dependencies: None

Eudev

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Gperf, Make, and Sed
Test suite depends on: None
Must be installed before: None
Optional dependencies: None

Expat

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, and Sed
Test suite depends on: None
Must be installed before: XML::Parser
Optional dependencies: None

Expect

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, and Tcl
Test suite depends on: None
Must be installed before: None
Optional dependencies: None

File

Installation depends on: Bash, Binutils, Bzip2, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, Xz, and Zlib
Test suite depends on: None
Must be installed before: None
Optional dependencies: None

Findutils

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: DejaGNU, Diffutils, and Expect
Must be installed before: None
Optional dependencies: None

Flex

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed, and Texinfo
Test suite depends on: Bison and Gawk
Must be installed before: IPRoute2, Kbd, and Man-DB
Optional dependencies: None

Gawk

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, MPFR, Patch, Readline, Sed, and Texinfo
Test suite depends on: Diffutils
Must be installed before: None
Optional dependencies: None

Gcc

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, and Texinfo
Test suite depends on: DejaGNU, Expect, and Shadow
Must be installed before: None
Optional dependencies: *GNAT* and *ISL*

GDBM

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make, and Sed
Test suite depends on: None
Must be installed before: None
Optional dependencies: None

Gettext

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: Diffutils, Perl, and Tcl
Must be installed before: Automake
Optional dependencies: None

Glibc

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Linux API Headers, Make, Perl, Python, Sed, and Texinfo
Test suite depends on: File
Must be installed before: None
Optional dependencies: None

GMP

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed, and Texinfo
Test suite depends on: None
Must be installed before: MPFR and GCC
Optional dependencies: None

Gperf

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, and Make
Test suite depends on: Diffutils and Expect
Must be installed before: None
Optional dependencies: None

Grep

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed, and Texinfo
Test suite depends on: Gawk
Must be installed before: Man-DB
Optional dependencies: Pcre

Groff

Installation depends on: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed, and Texinfo
Test suite depends on: No test suite available
Must be installed before: Man-DB and Perl
Optional dependencies: Ghostscript

GRUB

Installation depends on: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo, and Xz
Test suite depends on: None
Must be installed before: None
Optional dependencies: None

Gzip

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: Diffutils and Less
Must be installed before: Man-DB
Optional dependencies: None

Iana-Etc

Installation depends on: Coreutils, Gawk, and Make
Test suite depends on: No test suite available
Must be installed before: Perl
Optional dependencies: None

Inetutils

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo, and Zlib
Test suite depends on: No test suite available
Must be installed before: Tar
Optional dependencies: None

Intltool

Installation depends on: Bash, Gawk, Glibc, Make, Perl, Sed, and XML::Parser
Test suite depends on: Perl
Must be installed before: None
Optional dependencies: None

IProute2

Installation depends on: Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, and Linux API Headers
Test suite depends on: No test suite available
Must be installed before: None
Optional dependencies: None

Kbd

Installation depends on: Bash, Binutils, Bison, Check, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch, and Sed
Test suite depends on: No test suite available
Must be installed before: None
Optional dependencies: None

Kmod

Installation depends on: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Pkg-config, Sed, Xz-Utils, and Zlib
Test suite depends on: No test suite available
Must be installed before: Eudev
Optional dependencies: None

Less

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, and Sed
Test suite depends on: No test suite available
Must be installed before: Gzip
Optional dependencies: Pcre

Libcap

Installation depends on: Attr, Bash, Binutils, Coreutils, GCC, Glibc, Perl, Make, and Sed
Test suite depends on: No test suite available
Must be installed before: None
Optional dependencies: Linux-PAM

Libelf

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, and Make
Test suite depends on: No test suite available
Must be installed before: Linux Kernel
Optional dependencies: None

Libffi

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Make, and Sed
Test suite depends on: DejaGnu
Must be installed before: Python
Optional dependencies: None

Libpipeline

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: Check
Must be installed before: Man-DB
Optional dependencies: None

Libtool

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: Autoconf, Automake, and Findutils
Must be installed before: None
Optional dependencies: None

Linux Kernel

Installation depends on: Bash, Bc, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Libelf, Make, Ncurses, OpenSSL, Perl, and Sed
Test suite depends on: No test suite available
Must be installed before: None
Optional dependencies: None

M4

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: Diffutils
Must be installed before: Autoconf and Bison
Optional dependencies: libsigsegv

Make

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: Perl and Procps-ng
Must be installed before: None
Optional dependencies: None

Man-DB

Installation depends on: Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Sed, and Xz
Test suite depends on: Util-linux
Must be installed before: None
Optional dependencies: None

Man-Pages

Installation depends on: Bash, Coreutils, and Make
Test suite depends on: No test suite available
Must be installed before: None
Optional dependencies: None

Meson

Installation depends on: Ninja and Python
Test suite depends on: No test suite available
Must be installed before: Systemd
Optional dependencies: None

MPC

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed, and Texinfo
Test suite depends on: None
Must be installed before: GCC
Optional dependencies: None

MPFR

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed, and Texinfo
Test suite depends on: None
Must be installed before: Gawk and GCC
Optional dependencies: None

Ncurses

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch, and Sed
Test suite depends on: No test suite available
Must be installed before: Bash, GRUB, Inetutils, Less, Procps-ng, Psmisc, Readline, Texinfo, Util-linux, and Vim
Optional dependencies: None

Ninja

Installation depends on: Binutils, Coreutils, Gcc, and Python
Test suite depends on: None
Must be installed before: Meson
Optional dependencies: Asciidoc, Doxygen, Emacs, and re2c

Openssl

Installation depends on: Binutils, Coreutils, Gcc, Make, and Perl
Test suite depends on: None
Must be installed before: Linux
Optional dependencies: None

Patch

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, and Sed
Test suite depends on: Diffutils
Must be installed before: None
Optional dependencies: Ed

Perl

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Groff, Make, Sed, and Zlib
Test suite depends on: Iana-Etc and Procps-ng
Must be installed before: Autoconf
Optional dependencies: None

Pkg-config

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Popt, and Sed
Test suite depends on: None
Must be installed before: Kmod
Optional dependencies: None

Popt

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, and Make
Test suite depends on: Diffutils and Sed
Must be installed before: Pkg-config
Optional dependencies: None

Procps-ng

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Make, and Ncurses
Test suite depends on: DejaGNU
Must be installed before: None
Optional dependencies: None

Psmisc

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, and Sed
Test suite depends on: No test suite available
Must be installed before: None
Optional dependencies: None

Python

Installation depends on: Bash, Binutils, Coreutils, GCC, Gdbm, Gettext, Glibc, Grep, Libffi, Make, Ncurses, and Sed
Test suite depends on: GDB and Valgrind
Must be installed before: Ninja
Optional dependencies: Berkeley DB, OpenSSL, SQLite, and Tk

Readline

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, and Texinfo
Test suite depends on: No test suite available
Must be installed before: Bash and Gawk
Optional dependencies: None

Sed

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: Diffutils and Gawk
Must be installed before: E2fsprogs, File, Libtool, and Shadow
Optional dependencies: None

Shadow

Installation depends on: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, and Sed
Test suite depends on: No test suite available
Must be installed before: Coreutils
Optional dependencies: Cracklib, and PAM

Sysklogd

Installation depends on: Binutils, Coreutils, GCC, Glibc, Make, and Patch
Test suite depends on: No test suite available
Must be installed before: None
Optional dependencies: None

Systemd

Installation depends on: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Expat, Gawk, GCC, Glibc, Gperf, Grep, Intltool, Libcap, Meson, Sed, and Util-linux
Test suite depends on: None
Must be installed before: None
Optional dependencies: Many, see *BLFS systemd page*

Sysvinit

Installation depends on: Binutils, Coreutils, GCC, Glibc, Make, and Sed
Test suite depends on: No test suite available
Must be installed before: None
Optional dependencies: None

Tar

Installation depends on: Acl, Attr, Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed, and Texinfo
Test suite depends on: Autoconf, Diffutils, Findutils, Gawk, and Gzip
Must be installed before: None
Optional dependencies: None

Tcl

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, and Sed
Test suite depends on: None
Must be installed before: None
Optional dependencies: None

Texinfo

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, and Sed
Test suite depends on: None
Must be installed before: None
Optional dependencies: None

Util-linux

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Eudev, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, and Zlib
Test suite depends on: None
Must be installed before: None
Optional dependencies: *Libcap-ng*

Vim

Installation depends on: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, and Sed
Test suite depends on: None
Must be installed before: None
Optional dependencies: Xorg, GTK+2, LessTif, Python, Tcl, Ruby, and GPM

XML::Parser

Installation depends on: Bash, Binutils, Coreutils, Expat, GCC, Glibc, Make, and Perl
Test suite depends on: Perl
Must be installed before: Intltool
Optional dependencies: None

Xz

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, and Make
Test suite depends on: None
Must be installed before: Eudev, File, GRUB, Kmod, and Man-DB
Optional dependencies: None

Zlib

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Make, and Sed
Test suite depends on: None
Must be installed before: File, Kmod, Perl, and Util-linux
Optional dependencies: None

Zstd

Installation depends on: Binutils, Coreutils, GCC, Glibc, Gzip, Make, and Xz
Test suite depends on: None
Must be installed before: None
Optional dependencies: None

Appendix D. Boot and sysconfig scripts version-20191031

The scripts in this appendix are listed by the directory where they normally reside. The order is `/etc/rc.d/init.d`, `/etc/sysconfig`, `/etc/sysconfig/network-devices`, and `/etc/sysconfig/network-devices/services`. Within each section, the files are listed in the order they are normally called.

D.1. /etc/rc.d/init.d/rc

The `rc` script is the first script called by init and initiates the boot process.

```
#!/bin/bash
########################################################################
# Begin rc
#
# Description : Main Run Level Control Script
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#             : DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

. /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@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 /var/run/interactive ] && source /var/run/interactive

# Attempt to stop all services started by the previous runlevel,
# and killed in this runlevel
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]}
      prev_start=/etc/rc.d/rc$previous.d/S[0-9][0-9]$suffix
      sysinit_start=/etc/rc.d/rcS.d/S[0-9][0-9]$suffix

      if [ "${runlevel}" != "0" -a "${runlevel}" != "6" ]; then
         if [ ! -f ${prev_start} -a  ! -f ${sysinit_start} ]; then
            MSG="WARNING:\n\n${i} can't be "
            MSG="${MSG}executed because it was not "
            MSG="${MSG}not started in the previous "
            MSG="${MSG}runlevel (${previous})."
            log_warning_msg "$MSG"
            continue
         fi
      fi

      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 functions in this runlevel
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$runlevel.d/S[0-9][0-9]}
      stop=/etc/rc.d/rc$runlevel.d/K[0-9][0-9]$suffix
      prev_start=/etc/rc.d/rc$previous.d/S[0-9][0-9]$suffix

      [ -f ${prev_start} -a ! -f ${stop} ] && continue
   fi

   check_script_status
      if [ "${SCRIPT_STAT}" == "1" ]; then
         SCRIPT_STAT="0"
         continue
      fi

   case ${runlevel} in
      0|6)
         run ${i} stop
         ;;
      *)
         run ${i} start
         ;;
   esac

   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\"" > /var/run/interactive
else
    rm -f /var/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) responsibilty
        # 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) responsibilty
    # 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.
            rm -f "${pidfile}"

            # 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 "/var/run/${progname}.pid" ]; then
                rm -f "/var/run/${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 "/var/run/${progname}.pid" ]; then
            pidfile="/var/run/${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 "${base}" -a -e "/var/run/${base}.pid" ]; then
         /bin/echo -e "${WARNING}${1} is not running but" \
            "/var/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 failyure 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} | {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 or -{signal} or {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 : Mount proc, sysfs, and run
#
# 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 /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
# Description:         Mounts /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
# 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 /run/shm
      chmod 1777 /run/shm /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

      ln -sfn /run/shm /dev/shm

      (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 "${1}" in
   start)
      # Exit if there's no modules file or there are no
      # valid entries
      [ -r /etc/sysconfig/modules ]             || exit 0
      egrep -qv '^($|#)' /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 $local_fs
# Should-Stop:
# Default-Start:       S
# Default-Stop:        0 6
# Short-Description:   Mounts and unmounts swap partitions.
# Description:         Mounts and unmounts 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
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Checks local filesystems before mounting.
# Description:         Checks local filesystmes 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..."
      mount --all --test-opts no_netdev >/dev/null
      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 devies 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
      ;;

   *)
      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 /var/run, /var/lock, and
#                      optionally, /tmp.  cleanfs also creates /var/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

      > /var/run/utmp

      if grep -q '^utmp:' /etc/group ; then
         chmod 664 /var/run/utmp
         chgrp utmp /var/run/utmp
      fi

      (exit ${failed})
      evaluate_retval

      if egrep -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

is_true()
{
   [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ]
}

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:       3 4 5
# Default-Stop:        0 1 2 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

# Note: sysklogd is not started in runlevel 2 due to possible
# remote logging configurations

. /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 $syslog localnet swap
# Should-Start:        firewalld iptables nftables
# Required-Stop:       $local_fs $syslog localnet swap
# Should-Stop:         firewalld iptables nftables
# Default-Start:       3 4 5
# Default-Stop:        0 1 2 6
# Short-Description:   Starts and configures network interfaces.
# Description:         Starts and configures network interfaces.
# X-LFS-Provided-By:   LFS
### END INIT INFO

case "${1}" in
   start)
      # Start all network interfaces
      for file in /etc/sysconfig/ifconfig.*
      do
         interface=${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

         /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 "${1}" in
   stop)
      log_info_msg "Sending all processes the TERM signal..."
      killall5 -15
      error_value=${?}
```

```
      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
      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
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            reboot
# Required-Start:
# 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
   stop)
      log_info_msg "Restarting system..."
      reboot -d -f -i
      ;;

   *)
      echo "Usage: ${0} {stop}"
      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
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            halt
# Required-Start:
# 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
   stop)
      halt -d -f -i -p
      ;;
```

```
   *)
      echo "Usage: {stop}"
      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..."
      start_daemon fully_qualified_path
      ;;

   stop)
      log_info_msg "Stopping..."
      killproc fully_qualified_path
      ;;

   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 deivce
#               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 configuration rules

The rules in this appendix are listed for convenience. Installation is normally done via instructions in Section 6.78, “Eudev-3.2.9”.

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 Licenses

This book is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 2.0 License.

Computer instructions may be extracted from the book under the MIT License.

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