

Linux From Scratch

Table of Contents

<u>Linux From Scratch</u>	1
<u>Gerard Beekmans</u>	1
<u>Dedication</u>	2
<u>Preface</u>	6
<u>Who would want to read this book</u>	7
<u>Who would not want to read this book</u>	8
<u>Organization</u>	9
<u>Part I – Introduction</u>	9
<u>Part II – Installation of the LFS system</u>	9
<u>Part III – Appendixes</u>	9
<u>I. Part I – Introduction</u>	10
<u>Chapter 1. Introduction</u>	11
<u>Introduction</u>	12
<u>How things are going to be done</u>	13
<u>Book versions</u>	14
<u>HTTP Mirrors</u>	14
<u>FTP Mirrors</u>	14
<u>Acknowledgements</u>	15
<u>Changelog</u>	16
<u>Mailinglists and archives</u>	27
<u>lfs–discuss</u>	27
<u>lfs–apps</u>	27
<u>lfs–announce</u>	27
<u>alfs–discuss</u>	28
<u>alfs–docs</u>	28
<u>alfs–ipc</u>	28
<u>alfs–profile</u>	28
<u>How to subscribe?</u>	28
<u>How to unsubscribe?</u>	29
<u>Mail archives</u>	29
<u>Contact information</u>	30
<u>Chapter 2. Important information</u>	31

Table of Contents

About \$LFS	32
How to download the software	33
How to install the software	34
II. Part II – Installing the LFS system	36
Chapter 3. Packages you need to download	37
Chapter 4. Preparing a new partition	42
Introduction	43
Creating a new partition	44
Creating a ext2 file system on the new partition	45
Mounting the new partition	46
Creating directories	47
Chapter 5. Preparing the LFS system	48
Introduction	49
Installing Bash	50
Installation of Bash	50
Command explanations	50
Contents	51
Description	51
Installing Binutils	52
Installation of Binutils	52
Command explanations	52
Description	52
Description	52
ld	52
as	53
ar	53
nm	53
objcopy	53
objdump	53
ranlib	53
size	53
strings	53
strip	54
c++filt	54

Table of Contents

addr2line	54
nlmconv	54
Installing Bzip2.....	55
Installation of Bzip2	55
Command explanations	55
Contents	55
Description	55
Bzip2	56
Bunzip2	56
bzip2	56
bzip2recover	56
Installing Diffutils.....	57
Installation of Diffutils	57
Command explanations	57
Contents	57
Description	57
cmp and diff	57
diff3	57
sdiff	58
Installing Fileutils.....	59
Installation of Fileutils	59
Contents	59
Description	59
chgrp	59
chmod	59
chown	59
cp	59
dd	60
df	60
ls, dir and vdir	60
dircolors	60
du	60
install	60
ln	60
mkdir	60
mkfifo	61
mknod	61
mv	61
rm	61
rmdir	61
sync	61
touch	61
Installing GCC on the normal system if necessary.....	62
Installation of GCC on the normal system if necessary	62

Table of Contents

Command explanations	62
Contents	63
Description	63
Compiler	63
Pre-processor	63
C++ Library	63
Installing GCC on the LFS system.....	64
Installation of GCC on the LFS system	64
Creating necessary symlinks	64
Command explanations	64
Contents	65
Description	65
Compiler	65
Pre-processor	65
C++ Library	65
Installing Linux Kernel.....	66
Installation of Linux Kernel	66
Command explanations	66
Contents	67
Description	67
Installing Glibc.....	68
Installation of Glibc	68
Copying old NSS library files	69
Command explanations	70
Contents	70
Description	70
Installing Grep.....	71
Installation of Grep	71
Contents	71
Description	71
egrep	71
fgrep	71
grep	71
Installing Gzip.....	72
Installation of Gzip	72
Contents	72
Description	72
gunzip	72
gzexe	72
gzip	72
zcat	72
zcmp	73
zdiff	73

Table of Contents

zforce	73
zgrep	73
zmore	73
znew	73
Installing Make.....	74
Installation of Make	74
Contents	74
Description	74
Installing Sed.....	75
Installation of Sed	75
Contents	75
Description	75
Installing Shellutils.....	76
Installation of Sh–utils	76
Contents	76
Description	76
basename	76
chroot	76
date	76
dirname	76
echo	77
env	77
expr	77
factor	77
false	77
groups	77
hostid	77
hostname	77
id	77
logname	78
nice	78
nohup	78
pathchk	78
pinky	78
printenv	78
printf	78
pwd	78
seq	78
sleep	79
stty	79
su	79
tee	79
test	79
true	79
tty	79

Table of Contents

uname	79
uptime	79
users	80
who	80
whoami	80
yes	80
Installing Tar.....	81
Installation of Tar	81
Contents	81
Description	81
tar	81
rmt	81
Installing Textutils.....	82
Installation of Textutils	82
Contents	82
Description	82
cat	82
cksum	82
comm	82
csplit	82
cut	83
expand	83
fmt	83
fold	83
head	83
join	83
md5sum	83
nl	83
od	83
paste	84
pr	84
ptx	84
sort	84
split	84
sum	84
tac	84
tail	84
tr	84
tsort	85
unexpand	85
uniq	85
wc	85
Creating passwd and group files.....	86
Mounting \$LFS/proc file system.....	87

Table of Contents

<u>Chapter 6. Installing basic system software</u>	88
<u>Introduction</u>	89
<u>Debugging symbols and compiler optimizations</u>	90
<u>Creating \$LFS/root/.bash_profile</u>	92
<u>Entering the chroot'ed environment</u>	93
<u>Setting up locale</u>	94
<u>Creating device files</u>	95
<u>Installation of MAKEDEV</u>	95
<u>Creating the /dev entries</u>	95
<u>Command explanations</u>	95
<u>Contents</u>	96
<u>Description</u>	96
<u>Installing Ed</u>	97
<u>Installation of Ed</u>	97
<u>Contents</u>	97
<u>Description</u>	97
<u>Installing Patch</u>	98
<u>Installation of Patch</u>	98
<u>Contents</u>	98
<u>Description</u>	98
<u>Installing Findutils</u>	99
<u>Installing Findutils</u>	99
<u>Contents</u>	99
<u>Description</u>	99
<u>Find</u>	99
<u>Locate</u>	99
<u>Updatedb</u>	99
<u>Xargs</u>	100
<u>Installing Mawk</u>	101
<u>Installation of Mawk</u>	101
<u>Contents</u>	101
<u>Description</u>	101
<u>mawk</u>	101
<u>Installing Ncurses</u>	102
<u>Installation of Ncurses</u>	102
<u>Command explanations</u>	102
<u>Contents</u>	102

Table of Contents

Description	102
The libraries	102
Tic	102
Infocmp	103
clear	103
tput	103
toe	103
tset	103
Installing Vim.....	104
Installation of Vim	104
Contents	104
Description	104
ctags	104
etags	104
ex	104
gview	105
gvim	105
rgview	105
rgvim	105
rview	105
rvim	105
view	105
vim	105
vimtutor	105
xxd	106
Installing GCC.....	107
Installation of GCC	107
Contents	107
Description	107
Compiler	107
Pre-processor	107
C++ Library	107
Installing Bison.....	109
Installation of Bison	109
Command explanations	109
Contents	109
Description	110
Installing Less.....	111
Installation of Less	111
Contents	111
Description	111
Installing Groff.....	112
Installation of Groff	112

Table of Contents

Contents	112
Description	112
addftinfo	112
afmtodit	112
eqn	112
grodvi	112
groff	113
grog	113
grohtml	113
grolj4	113
grops	113
grotty	113
hpfodit	113
indxbib	113
lkbib	113
lookbib	114
neqn	114
nroff	114
pfbtops	114
pic	114
psbb	114
refer	114
soelim	114
tbl	115
tfmtodit	115
troff	115
Installing Man.....	116
Installation of Man	116
Command explanations	116
Contents	116
Description	116
man	116
apropos	116
whatis	117
makewhatis	117
Installing Perl.....	118
Installation of Perl	118
Contents	118
Description	118
Installing M4.....	119
Installation of M4	119
Contents	119
Description	120
Installing Texinfo.....	121

Table of Contents

Installation of Texinfo	121
Contents	121
Description	121
info	121
install-info	121
makeinfo	121
texi2dvi	121
texindex	122
Installing Autoconf.....	123
Installation of Autoconf	123
Contents	123
Description	123
autoconf	123
autoheader	123
autoreconf	123
autoscan	123
autoupdate	124
ifnames	124
Installing Automake.....	125
Installation of Automake	125
Contents	125
Description	125
aclocal	125
automake	125
Installing Bash.....	126
Installation of Bash	126
Contents	126
Description	126
Installing Flex.....	127
Installation of Flex	127
Contents	127
Description	127
Installing File.....	128
Installation of File	128
Contents	128
Description	128
Installing Libtool.....	129
Installation of Libtool	129
Contents	129
Description	129
libtool	129
libtoolize	129

Table of Contents

ltdl library	129
Installing Bin86.....	130
Installation of Bin86	130
Contents	130
Description	130
as86	130
as86_encap	130
ld86	130
objdump86	130
nm86	131
size86	131
Installing Binutils.....	132
Installation of Binutils	132
Description	132
Description	132
ld	132
as	132
ar	132
nm	132
objcopy	133
objdump	133
ranlib	133
size	133
strings	133
strip	133
c++filt	133
addr2line	134
nlmconv	134
Installing Bzip2.....	135
Installation of Bzip2	135
Command explanations	135
Contents	135
Description	136
Bzip2	136
Bunzip2	136
bzip2	136
bzip2recover	136
Installing Gettext.....	137
Installation of Gettext	137
Contents	137
Description	137
gettext	137
Installing Consoletools.....	138

Table of Contents

Installation of Console–tools	138
Contents	138
Description	138
charset	138
chvt	138
codepage	138
consolechars	139
deallocvt	139
dumpkeys	139
fgconsole	139
fix_bs_and_del	139
font2psf	139
getkeycodes	139
kbd_mode	139
loadkeys	139
loadunimap	140
mapscrn	140
mk_modmap	140
openvt	140
psfaddtable	140
psfgettable	140
psfstriptime	140
resizecons	140
saveunimap	140
screendump	141
setfont	141
setkeycodes	141
setleds	141
setmetamode	141
setvesablank	141
showcfont	141
showkey	141
splitfont	141
unicode_start	142
unicode_stop	142
vcstime	142
vt-is-UTF8	142
writevt	142
Installing Consoledata.....	143
Installation of Console–data	143
Contents	143
Installing Diffutils.....	144
Installation of Diffutils	144
Contents	144
Description	144
cmp and diff	144

Table of Contents

diff3	144
sdiff	144
Installing E2fsprogs.....	145
Installation of E2fsprogs	145
Contents	145
Description	145
chattr	145
lsattr	145
uuidgen	145
badblocks	146
debugfs	146
dumpe2fs	146
e2fsck and fsck.ext2	146
e2label	146
fsck	146
mke2fs and mkfs.ext2	146
mklost+found	146
tune2fs	147
Installing Fileutils.....	148
Installation of Fileutils	148
Contents	148
Description	148
chgrp	148
chmod	148
chown	148
cp	148
dd	149
df	149
ls_dir and vdir	149
dircolors	149
du	149
install	149
ln	149
mkdir	149
mkfifo	150
mknod	150
mv	150
rm	150
rmdir	150
sync	150
touch	150
Installing Grep.....	151
Installation of Grep	151
Contents	151
Description	151

Table of Contents

egrep	151
fgrep	151
grep	151
Installing Gzip.....	152
Installation of Gzip	152
Contents	152
Description	152
gunzip	152
gzexe	152
gzip	152
zcat	153
zcmp	153
zdiff	153
zforce	153
zgrep	153
zmore	153
znew	153
Installing Ldso.....	154
Installation of Ld.so	154
Contents	154
Description	154
ldconfig	154
ldd	155
Installing Lilo.....	156
Installation of Lilo	156
Contents	156
Description	156
Installing Make.....	157
Installation of Make	157
Contents	157
Description	157
Installing Modutils.....	158
Installation of Modutils	158
Contents	158
Description	158
depmod	158
genksyms	158
insmod	158
insmod ksymbols clean	158
kerneld	159
kernelversion	159
ksyms	159
lsmod	159

Table of Contents

modinfo	159
modprobe	159
rmmod	159
Installing Procinfo.....	160
Installation of Procinfo	160
Contents	160
Description	160
Installing Procps.....	161
Installation of Procps	161
Command explanations	161
Contents	161
Description	161
free	161
kill	161
oldps and ps	162
skill	162
snice	162
sysctl	162
tload	162
top	162
uptime	162
vmstat	162
w	162
watch	163
Installing Psmisc.....	164
Installation of Psmisc	164
Contents	164
Description	164
fuser	164
killall	164
pstree	164
Installing Sed.....	165
Installation of Sed	165
Contents	165
Description	165
Installing Shellutils.....	166
Installation of Sh–utils	166
Contents	166
Description	166
basename	166
chroot	166
date	166
dirname	166

Table of Contents

echo	167
env	167
expr	167
factor	167
false	167
groups	167
hostid	167
hostname	167
id	167
logname	168
nice	168
nohup	168
pathchk	168
pinky	168
printenv	168
printf	168
pwd	168
seq	168
sleep	169
stty	169
su	169
tee	169
test	169
true	169
tty	169
uname	169
uptime	169
users	170
who	170
whoami	170
yes	170
Installing Shadowpwd	171
Installation of Shadow Password Suite	171
Command explanations	171
Contents	171
Description	171
chage	171
chfn	171
chsh	172
expiry	172
faillog	172
gpasswd	172
lastlog	172
login	172
newgrp	172
passwd	172
sg	172

Table of Contents

su	173
chpasswd	173
dpasswd	173
groupadd	173
groupdel	173
groupmod	173
grpck	173
grpconv	173
grpunconv	174
logoutd	174
mkpasswd	174
newusers	174
pwck	174
pwconv	174
pwunconv	174
useradd	174
userdel	174
usermod	175
vipw and vigr	175
Installing Sysklogd.....	176
Installation of Sysklogd	176
Contents	176
Description	176
klogd	176
syslogd	176
Installing Sysvinit.....	177
Installation of Sysvinit	177
Contents	177
Description	177
pidof	177
last	177
lastb	177
mesg	177
utmpdump	178
wall	178
halt	178
init	178
killall5	178
poweroff	178
reboot	178
runlevel	178
shutdown	179
sulogin	179
telinit	179
Installing Tar.....	180

Table of Contents

Installation of Tar	180
Contents	180
Description	180
tar	180
rmt	180
Installing Textutils.....	181
Installation of Textutils	181
Contents	181
Description	181
cat	181
cksum	181
comm	181
csplit	181
cut	182
expand	182
fmt	182
fold	182
head	182
join	182
md5sum	182
nl	182
od	182
paste	183
pr	183
ptx	183
sort	183
split	183
sum	183
tac	183
tail	183
tr	183
tsort	184
unexpand	184
uniq	184
wc	184
Installing Util-linux.....	185
Installation of Util-Linux	185
Command explanations	185
Contents	186
Description	186
arch	186
dmesg	186
kill	186
more	186
mount	186
umount	186

Table of Contents

<u>agetty</u>	186
<u>blockdev</u>	187
<u>fdisk</u>	187
<u>ctrlaltdel</u>	187
<u>elvtune</u>	187
<u>fdisk</u>	187
<u>fsck.minix</u>	187
<u>hwclock</u>	187
<u>kbdrate</u>	187
<u>losetup</u>	187
<u>mkfs</u>	188
<u>mkfs.bfs</u>	188
<u>mkfs.minix</u>	188
<u>mkswap</u>	188
<u>sfdisk</u>	188
<u>swapoff</u>	188
<u>swapon</u>	188
<u>cal</u>	188
<u>chkdupexe</u>	188
<u>col</u>	189
<u>colert</u>	189
<u>colrm</u>	189
<u>column</u>	189
<u>cytune</u>	189
<u>ddate</u>	189
<u>fdformat</u>	189
<u>getopt</u>	189
<u>hexdump</u>	189
<u>ipcrm</u>	190
<u>ipcs</u>	190
<u>logger</u>	190
<u>look</u>	190
<u>mcookie</u>	190
<u>namei</u>	190
<u>rename</u>	190
<u>renice</u>	190
<u>rev</u>	190
<u>script</u>	191
<u>setfdprm</u>	191
<u>setsid</u>	191
<u>setterm</u>	191
<u>ul</u>	191
<u>whereis</u>	191
<u>write</u>	191
<u>ramsize</u>	191
<u>rdev</u>	191
<u>readprofile</u>	192
<u>rootflags</u>	192

Table of Contents

swapdev	192
tunelp	192
vidmode	192
Installing Man–pages.....	193
Installation of Man–pages	193
Command explanations	193
Contents	193
Description	193
Removing old NSS library files.....	194
Configuring essential software.....	195
Configuring Vim	195
Configuring Glibc	195
Configuring Dynamic Loader	196
Configuring Lilo	197
Configuring Sysklogd	198
Configuring Shadow Password Suite	198
Configuring Sysvinit	198
Creating the /var/run/utmp, /var/log/wtmp and /var/log/btmp files	199
Creating root password	200
Chapter 7. Creating system boot scripts.....	201
Introduction.....	202
Creating directories.....	203
Creating the rc script.....	204
Creating the rcS script.....	207
Creating the functions script.....	208
Creating the checkfs script.....	213
Creating the halt script.....	215
Creating the loadkeys script.....	216
Creating the mountfs script.....	217
Creating the reboot script.....	218
Creating the sendsignals script.....	219
Creating the setclock script.....	220

Table of Contents

Creating the /etc/sysconfig/clock file	220
Creating the syslogd script	222
Creating the template script	224
Creating the umountfs script	225
Setting up symlinks and permissions	226
Creating the /etc/fstab file	227
Chapter 8. Making the LFS system bootable	228
Introduction	229
Installing a kernel	230
Adding an entry to LILO	231
Rebooting the system	232
Chapter 9. Setting up basic networking	233
Introduction	234
Installing network software	235
Installing Netkit-base	235
Installing Net-tools	235
Creating network boot scripts	236
Creating the /etc/init.d/localnet bootsript	236
Setting up permissions and symlink	236
Creating the /etc/sysconfig/network file	237
Creating the /etc/hosts file	237
Creating the /etc/init.d/ethnet script	238
Adding default gateway to /etc/sysconfig/network	239
Creating NIC configuration files	240
Setting up permissions and symlink	240
III. Part III – Appendixes	242
Appendix A. Package descriptions	243
Introduction	244
Glibc	245
Contents	245

Table of Contents

Description	245
Linux kernel.....	246
Contents	246
Description	246
Ed.....	247
Contents	247
Description	247
Patch.....	248
Contents	248
Description	248
GCC.....	249
Contents	249
Description	249
Compiler	249
Pre-processor	249
C++ Library	249
Bison.....	250
Contents	250
Description	250
Mawk.....	251
Contents	251
Description	251
mawk	251
Findutils.....	252
Contents	252
Description	252
Find	252
Locate	252
Updatedb	252
Xargs	252
Ncurses.....	253
Contents	253
Description	253
The libraries	253
Tic	253
Infocmp	253
clear	253
tput	253
toe	254
tset	254

Table of Contents

<u>Less</u>	255
<u>Contents</u>	255
<u>Description</u>	255
<u>Groff</u>	256
<u>Contents</u>	256
<u>Description</u>	256
<u>addftinfo</u>	256
<u>afmtodit</u>	256
<u>eqn</u>	256
<u>grodvi</u>	256
<u>groff</u>	256
<u>grog</u>	256
<u>grohtml</u>	257
<u>grolj4</u>	257
<u>grops</u>	257
<u>grotty</u>	257
<u>hpftodit</u>	257
<u>indxbib</u>	257
<u>lkbib</u>	257
<u>lookbib</u>	257
<u>neqn</u>	258
<u>nroff</u>	258
<u>pfbtops</u>	258
<u>pic</u>	258
<u>psbb</u>	258
<u>refer</u>	258
<u>soelim</u>	258
<u>tbl</u>	258
<u>tfmtodit</u>	259
<u>troff</u>	259
<u>Man</u>	260
<u>Contents</u>	260
<u>Description</u>	260
<u>man</u>	260
<u>apropos</u>	260
<u>whatis</u>	260
<u>makewhatis</u>	260
<u>Perl</u>	261
<u>Contents</u>	261
<u>Description</u>	261
<u>M4</u>	262
<u>Contents</u>	262
<u>Description</u>	262

Table of Contents

<u>Texinfo</u>	263
<u>Contents</u>	263
<u>Description</u>	263
<u>info</u>	263
<u>install-info</u>	263
<u>makeinfo</u>	263
<u>texi2dvi</u>	263
<u>texindex</u>	263
<u>Autoconf</u>	264
<u>Contents</u>	264
<u>Description</u>	264
<u>autoconf</u>	264
<u>autoheader</u>	264
<u>autoreconf</u>	264
<u>autoscan</u>	264
<u>autoupdate</u>	264
<u>ifnames</u>	264
<u>Automake</u>	266
<u>Contents</u>	266
<u>Description</u>	266
<u>aclocal</u>	266
<u>automake</u>	266
<u>Bash</u>	267
<u>Contents</u>	267
<u>Description</u>	267
<u>Flex</u>	268
<u>Contents</u>	268
<u>Description</u>	268
<u>Binutils</u>	269
<u>Description</u>	269
<u>Description</u>	269
<u>ld</u>	269
<u>as</u>	269
<u>ar</u>	269
<u>nm</u>	269
<u>objcopy</u>	269
<u>objdump</u>	269
<u>ranlib</u>	270
<u>size</u>	270
<u>strings</u>	270
<u>strip</u>	270
<u>c++filt</u>	270
<u>addr2line</u>	270

Table of Contents

nlmconv	271
Bzip2.....	272
Contents	272
Description	272
Bzip2	272
Bunzip2	272
bzipcat	272
bzip2recover	272
Diffutils.....	273
Contents	273
Description	273
cmp and diff	273
diff3	273
sdiff	273
E2fsprogs.....	274
Contents	274
Description	274
chattr	274
lsattr	274
uuidgen	274
badblocks	274
debugfs	274
dumpe2fs	274
e2fsck and fsck.ext2	275
e2label	275
fsck	275
mke2fs and mkfs.ext2	275
mklost+found	275
tune2fs	275
File.....	276
Contents	276
Description	276
Fileutils.....	277
Contents	277
Description	277
chgrp	277
chmod	277
chown	277
cp	277
dd	277
df	277
ls, dir and vdir	278
dircolors	278

Table of Contents

du	278
install	278
ln	278
mkdir	278
mkfifo	278
mknod	278
mv	279
rm	279
rmdir	279
sync	279
touch	279
Gettext.....	280
Contents	280
Description	280
gettext	280
Grep.....	281
Contents	281
Description	281
egrep	281
fgrep	281
grep	281
Gzip.....	282
Contents	282
Description	282
gunzip	282
gzexe	282
gzip	282
zcat	282
zcmp	282
zdiff	282
zforce	282
zgrep	283
zmore	283
znew	283
Ld.so.....	284
Contents	284
Description	284
ldconfig	284
ldd	284
Libtool.....	285
Contents	285
Description	285
libtool	285

Table of Contents

libtoolize	285
ltdl library	285
Bin86.....	286
Contents	286
Description	286
as86	286
as86_encap	286
ld86	286
objdump86	286
nm86	286
size86	286
Lilo.....	287
Contents	287
Description	287
Make.....	288
Contents	288
Description	288
Shellutils.....	289
Contents	289
Description	289
basename	289
chroot	289
date	289
dirname	289
echo	289
env	289
expr	289
factor	290
false	290
groups	290
hostid	290
hostname	290
id	290
logname	290
nice	290
nohup	290
pathchk	291
pinky	291
printenv	291
printf	291
pwd	291
seq	291
sleep	291
stty	291

Table of Contents

su	291
tee	292
test	292
true	292
tty	292
uname	292
uptime	292
users	292
who	292
whoami	292
yes	293
Shadow Password Suite.....	294
Contents	294
Description	294
chage	294
chfn	294
chsh	294
expiry	294
faillog	294
gpasswd	294
lastlog	294
login	295
newgrp	295
passwd	295
sg	295
su	295
chpasswd	295
dpasswd	295
groupadd	295
groupdel	296
groupmod	296
grpck	296
grpconv	296
grpunconv	296
logoutd	296
mkpasswd	296
newusers	296
pwck	296
pwconv	297
pwunconv	297
useradd	297
userdel	297
usermod	297
vipw and vigr	297
Modutils.....	298
Contents	298

Table of Contents

<u>Description</u>	298
<u>depmod</u>	298
<u>genksyms</u>	298
<u>insmod</u>	298
<u>insmod ksymoops clean</u>	298
<u>kerneld</u>	298
<u>kernelversion</u>	298
<u>ksyms</u>	298
<u>lsmod</u>	299
<u>modinfo</u>	299
<u>modprobe</u>	299
<u>rmmod</u>	299
<u>Procinfo</u>.....	300
<u>Contents</u>	300
<u>Description</u>	300
<u>Procps</u>.....	301
<u>Contents</u>	301
<u>Description</u>	301
<u>free</u>	301
<u>kill</u>	301
<u>oldps and ps</u>	301
<u>skill</u>	301
<u>snice</u>	301
<u>sysctl</u>	301
<u>tload</u>	301
<u>top</u>	302
<u>uptime</u>	302
<u>vmstat</u>	302
<u>w</u>	302
<u>watch</u>	302
<u>Vim</u>.....	303
<u>Contents</u>	303
<u>Description</u>	303
<u>ctags</u>	303
<u>etags</u>	303
<u>ex</u>	303
<u>gview</u>	303
<u>gvim</u>	303
<u>rgview</u>	303
<u>rgvim</u>	303
<u>rview</u>	304
<u>rvim</u>	304
<u>view</u>	304
<u>vim</u>	304
<u>vimtutor</u>	304

Table of Contents

xxd	304
Psmisc.....	305
Contents	305
Description	305
fuser	305
killall	305
pstree	305
Sed.....	306
Contents	306
Description	306
Syslogd.....	307
Contents	307
Description	307
klogd	307
syslogd	307
Sysvinit.....	308
Contents	308
Description	308
pidof	308
last	308
lastb	308
mesg	308
utmpdump	308
wall	308
halt	309
init	309
killall5	309
poweroff	309
reboot	309
runlevel	309
shutdown	309
sulogin	309
telinit	310
Tar.....	311
Contents	311
Description	311
tar	311
rmt	311
Textutils.....	312
Contents	312
Description	312
cat	312

Table of Contents

cksum	312
comm	312
csplit	312
cut	312
expand	312
fmt	312
fold	313
head	313
join	313
md5sum	313
nl	313
od	313
paste	313
pr	313
ptx	313
sort	314
split	314
sum	314
tac	314
tail	314
tr	314
tsort	314
unexpand	314
uniq	314
wc	315
Util Linux	316
Contents	316
Description	316
arch	316
dmesg	316
kill	316
more	316
mount	316
umount	316
agetty	317
blockdev	317
cfdisk	317
ctrlaltdel	317
elvtune	317
fdisk	317
fsck.minix	317
hwclock	317
kbdrate	317
losetup	318
mkfs	318
mkfs.bfs	318
mkfs.minix	318

Table of Contents

mkswap	318
sfdisk	318
swapoff	318
swapon	318
cal	318
chkdupexe	319
col	319
colcrt	319
colrm	319
column	319
cytune	319
ddate	319
fdformat	319
getopt	319
hexdump	320
iperm	320
ipcs	320
logger	320
look	320
mcookie	320
namei	320
rename	320
renice	320
rev	321
script	321
setfdprm	321
setsid	321
setterm	321
ul	321
whereis	321
write	321
ramsize	321
rdev	322
readprofile	322
rootflags	322
swapdev	322
tunelp	322
vidmode	322
Console-tools	323
Contents	323
Description	323
charset	323
chvt	323
codepage	323
consolechars	323
deallocvt	323
dumpkeys	323

Table of Contents

fgconsole	324
fix_bs_and_del	324
font2psf	324
getkeycodes	324
kbd_mode	324
loadkeys	324
loadunimap	324
mapscrn	324
mk_modmap	324
openvt	325
psfaddtable	325
psfgettable	325
psfstriptime	325
resizecons	325
saveunimap	325
screendump	325
setfont	325
setkeycodes	325
setleds	326
setmetamode	326
setvesablank	326
showfont	326
showkey	326
splitfont	326
unicode_start	326
unicode_stop	326
vcstime	326
vt-is-UTF8	327
writevt	327
Console-data.....	328
Contents	328
Man-pages.....	329
Contents	329
Description	329
Appendix B. Resources.....	330
Introduction.....	331
Books.....	332
HOWTOs and Guides.....	333
Other.....	334
Appendix C. Official download locations.....	335

Linux From Scratch

Gerard Beekmans

Copyright © 1999, 2000 by Gerard Beekmans

This book describes the process of creating your own Linux system from scratch from an already installed Linux distribution, using nothing but the sources of software that are needed.

Copyright (c) 1999–2000, Gerard Beekmans

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of LinuxFromScratch nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Dedication

This book is dedicated to my loving and supportive wife *Beverly Beekmans*.

Table of Contents

[Preface](#)

[Who would want to read this book](#)

[Who would not want to read this book](#)

[Organization](#)

[Part I – Introduction](#)

[Part II – Installation of the LFS system](#)

[Part III – Appendixes](#)

I. [Part I – Introduction](#)

1. [Introduction](#)

[Introduction](#)

[How things are going to be done](#)

[Book versions](#)

[Acknowledgements](#)

[Changelog](#)

[Mailinglists and archives](#)

[Contact information](#)

2. [Important information](#)

[About \\$LFS](#)

[How to download the software](#)

[How to install the software](#)

II. [Part II – Installing the LFS system](#)

3. [Packages you need to download](#)

4. [Preparing a new partition](#)

[Introduction](#)

[Creating a new partition](#)

[Creating a ext2 file system on the new partition](#)

[Mounting the new partition](#)

[Creating directories](#)

5. [Preparing the LFS system](#)

[Introduction](#)

[Installing Bash](#)

[Installing Binutils](#)

[Installing Bzip2](#)

[Installing Diffutils](#)

[Installing Fileutils](#)

[Installing GCC on the normal system if necessary](#)

[Installing GCC on the LFS system](#)

[Installing Linux Kernel](#)

[Installing Glibc](#)

[Installing Grep](#)

[Installing Gzip](#)

[Installing Make](#)

[Installing Sed](#)

[Installing Shellutils](#)

[Installing Tar](#)

[Installing Textutils](#)

[Creating passwd and group files](#)

[Mounting \\$LFS/proc file system](#)

6. [Installing basic system software](#)

[Introduction](#)

[Debugging symbols and compiler optimizations](#)

[Creating \\$LFS/root/.bash_profile](#)

[Entering the chroot'ed environment](#)

[Setting up locale](#)

[Creating device files](#)

[Installing Ed](#)

[Installing Patch](#)

[Installing Findutils](#)

[Installing Mawk](#)

[Installing Ncurses](#)

[Installing Vim](#)

[Installing GCC](#)

[Installing Bison](#)

[Installing Less](#)

[Installing Groff](#)

[Installing Man](#)

[Installing Perl](#)

[Installing M4](#)

[Installing Texinfo](#)

[Installing Autoconf](#)

[Installing Automake](#)

[Installing Bash](#)

[Installing Flex](#)

[Installing File](#)

[Installing Libtool](#)

[Installing Bin86](#)

[Installing Binutils](#)

[Installing Bzip2](#)

[Installing Gettext](#)

[Installing Consoletools](#)

[Installing Consoledata](#)

[Installing Diffutils](#)

[Installing E2fsprogs](#)

[Installing Fileutils](#)

[Installing Grep](#)

[Installing Gzip](#)

[Installing Ldso](#)

[Installing Lilo](#)

[Installing Make](#)

[Installing Modutils](#)

[Installing Procinfo](#)

[Installing Procps](#)

[Installing Psmisc](#)

[Installing Sed](#)

[Installing Shellutils](#)

[Installing Shadowpwd](#)

[Installing Sysklogd](#)
[Installing Sysvinit](#)
[Installing Tar](#)
[Installing Textutils](#)
[Installing Utilinux](#)
[Installing Man–pages](#)
[Removing old NSS library files](#)
[Configuring essential software](#)

7. [Creating system boot scripts](#)

[Introduction](#)
[Creating directories](#)
[Creating the rc script](#)
[Creating the rcS script](#)
[Creating the functions script](#)
[Creating the checkfs script](#)
[Creating the halt script](#)
[Creating the loadkeys script](#)
[Creating the mountfs script](#)
[Creating the reboot script](#)
[Creating the sendsignals script](#)
[Creating the setclock script](#)
[Creating the sysklogd script](#)
[Creating the template script](#)
[Creating the umountfs script](#)
[Setting up symlinks and permissions](#)
[Creating the /etc/fstab file](#)

8. [Making the LFS system bootable](#)

[Introduction](#)
[Installing a kernel](#)
[Adding an entry to LILO](#)
[Rebooting the system](#)

9. [Setting up basic networking](#)

[Introduction](#)
[Installing network software](#)
[Creating network boot scripts](#)

III. [Part III – Appendixes](#)

A. [Package descriptions](#)

[Introduction](#)
[Glibc](#)
[Linux kernel](#)
[Ed](#)
[Patch](#)
[GCC](#)
[Bison](#)
[Mawk](#)
[Findutils](#)
[Ncurses](#)
[Less](#)
[Groff](#)
[Man](#)
[Perl](#)

[M4](#)
[Texinfo](#)
[Autoconf](#)
[Automake](#)
[Bash](#)
[Flex](#)
[Binutils](#)
[Bzip2](#)
[Diffutils](#)
[E2fsprogs](#)
[File](#)
[Fileutils](#)
[Gettext](#)
[Grep](#)
[Gzip](#)
[Ld.so](#)
[Libtool](#)
[Bin86](#)
[Lilo](#)
[Make](#)
[Shellutils](#)
[Shadow Password Suite](#)
[Modutils](#)
[Procinfa](#)
[Procps](#)
[Vim](#)
[Psmisc](#)
[Sed](#)
[Sysklogd](#)
[Sysvinit](#)
[Tar](#)
[Textutils](#)
[Util Linux](#)
[Console-tools](#)
[Console-data](#)
[Man-pages](#)

B. [Resources](#)

[Introduction](#)
[Books](#)
[HOWTOs and Guides](#)
[Other](#)

C. [Official download locations](#)

Preface

Who would want to read this book

This book is intended for Linux users who want to learn more about the inner workings of Linux and how the various pieces of the Operating System fit together. This book will guide you step-by-step in creating your own custom build Linux system from scratch, using the source code of the software that we need.

This book is also intended for Linux users who want to get away from the existing commercial and free distributions that are often too bloated. Using existing distributions also forces you to use the file system structure, boot script structure, etc. that they choose to use. With this book you can create your own structures and methods in exactly the way you like them (which can be based on the ones this book provides)

Also, if you have security concerns, you don't want to rely on pre-compiled packages. So instead, you want to compile all programs from scratch and install them yourself. That could be another reason why you would want to build a custom made Linux system.

Those are just a few out of many reasons why people want to build their own Linux system. If you're one of those people, this book is meant for you.

Who would not want to read this book

Users who don't want to build an entire Linux system from scratch probably don't want to read this book. If you, however, do want to learn more about what happens behind the scenes, in particular what happens between turning on your computer and seeing the command prompt, you want to read the "From Power Up To Bash Prompt" (P2B) HOWTO. This HOWTO builds a bare system, in a similar way as this book does, but it focusses more on just installing a bootable system instead of a complete system.

To decide whether you want to read this book or the P2B HOWTO, you could ask yourself this question: Is my main objective to get a working Linux system that I'm going to build myself and along the way learn and learn what every component of a system is for, or is just the learning part your main objective. If you want to build and learn, read this book. If you just want to learn, then the P2B HOWTO is probably better material to read.

The "From Power Up To Bash Prompt" HOWTO can be downloaded from <http://learning.taslug.org.au/power2bash>

Organization

This book is divided into the following parts. Although there is a lot of duplicate information in certain parts, it's the easiest way to read it and not to mention the easiest way for me to maintain the book.

Part I – Introduction

Part One gives you general information about this book (versions, where to get it, changelog, mailinglists and how to get in touch with me). It also explains a few important aspects you really want and need to read before you start building an LFS system.

Part II – Installation of the LFS system

Part Two guides you through the installation of the LFS system which will be the foundation for the rest of the system. Whatever you choose to do with your brand new LFS system, it will be built on the foundation that's installed in this part.

Part III – Appendixes

Part Three contains various Appendixes.

I. Part I – Introduction

Table of Contents

1. [Introduction](#)
 2. [Important information](#)
-

Chapter 1. Introduction

Introduction

Having used a number of different Linux distributions, I was never fully satisfied with any of those. I didn't like the way the bootscripts were arranged, or I didn't like the way certain programs were configured by default and more of those things. I came to realize that when I want to be totally satisfied with a Linux system, I have to build my own Linux system from scratch, ideally only using the source code. Not using pre-compiled packages of any kind. No help from some sort of cdrom or bootdisk that would install some basic utilities. You would use your current Linux system and use that one to build your own.

This, at one time, wild idea seemed very difficult and at times almost impossible. The reason for most problems were due to my lack of knowledge about certain programs and procedures. After sorting out all kinds of dependency problems, compilation problems, etcetera, a custom built Linux system was created and fully operational. I called this system an LFS system, which stands for LinuxFromScratch.

How things are going to be done

We are going to build the LFS system by using an already installed Linux distribution such as Debian, SuSe, Slackware, Mandrake, RedHat, etc. You don't need to have any kind of bootdisk. We will use an existing Linux system as the base (since we need a compiler, linker, text editor and other tools).

If you don't have Linux installed yet, you won't be able to put this book to use right away. I suggest you first install a Linux distribution. It really doesn't matter which one you install. It also doesn't need to be the latest version, though it shouldn't be a too old one. If it is about a year old or newer it should do just fine. You will save yourself a lot of trouble if your normal system uses glibc-2.1 or newer. Libc5 isn't supported by this book, though it isn't impossible to use a libc5 system if you have no choice.

Book versions

This is LFS-BOOK-INTEL version 2.4.3 version dated November 21st, 2000. If this version is older than a month you definitely want to take a look at our website and download a newer version.

Below you will find a list of our current HTTP and FTP mirror sites as of November 14th, 2000. This list might not be accurate anymore. For the latest info check our website at <http://www.linuxfromscratch.org>

HTTP Mirrors

- Columbus, Ohio, United States – <http://www.linuxfromscratch.org/intro/>
 - United States – <http://lfs.sourceforge.net/intro/>
 - Canmore, Alberta, Canada – <http://www.ca.linuxfromscratch.org/intro/>
 - Braunschweig, Niedersachsen, Germany – <http://www.de.linuxfromscratch.org/intro/>
 - Australia (accessible from within AU/NZ only) – <http://lfs.mirror.aarnet.edu.au/intro/>
-

FTP Mirrors

- Columbus, Ohio, USA – <ftp://packages.linuxfromscratch.org>
 - Canmore, Alberta, Canada [FTP Interface to FTP archive] – <ftp://ftp.ca.linuxfromscratch.org>
 - Canmore, Alberta, Canada [HTTP Interface to FTP archive] – <http://ftp.ca.linuxfromscratch.org>
 - Mainz, Germany, Europe – <ftp://ftp.linux-provider.net/lfs/>
 - Australia (accessible from within AU/NZ only) – <ftp://mirror.aarnet.edu.au/pub/lfs/>
-

Acknowledgements

I would like to thank the following people and organizations for their contributions towards the LinuxFromScratch project:

- [Bryan Dumm](#) for providing the hardware to run linuxfromscratch.org and for providing <http://www.bcpub.com> as the lfs.bcpub.com mirror
 - [Jan Niemann](#) for providing <http://helga.lk.etc.tu-bs.de> as the 134.169.139.209 mirror
 - [Johan Lenglet](#) for providing <http://www.linuxien.com> as the www.linuxien.com/lfs mirror
 - [David Downey](#) for providing <http://www.codecastle.com> as the lfs.codecastle.com mirror
 - [Michael Peters](#) for contributing the Apple PowerPC modifications
 - [VA Linux Systems](#) who, on behalf of [Linux.com](#), donated a VA Linux 420 (formerly StartX SP2) workstation towards this project
 - [Jesse Tie Ten Quee](#) who donated a Yamaha CDRW 8824E CD-RW.
 - [Jesse Tie Ten Quee](#) for providing quasar.highos.com as the www.ca.linuxfromscratch.org mirror.
 - [DREAMWVR.COM](#) for their ongoing sponsorship.
 - [O'Reilly](#) for donating books on SQL and PHP.
 - Robert Briggs for donating the linuxfromscratch.org and linuxfromscratch.com domain names.
 - Countless other people from the various LFS mailinglists who are making this book happen by making suggestions, testing and submitting bug reports.
-

Changelog

If, for example, a change is listed for chapter 5 it (usually) means the same change has been made in the chapters for the other architectures.

j.4.3 November 21st, 2000

- The LFS FTP archive has been moved to a new server which is reachable under the name `packages.linuxfromscratch.org`. The reason for the move is that this new server sits on a link with a lot more bandwidth to spare.
- Instead of having the reader create files by running vim or some editor, the reader can now simply copy and paste a command that creates the file in the form of `"cat > outputfile << EOF"` followed by the text to put in the file and when a single line containing EOF is read by cat, it stops reading and writes the file (not including the EOF). This will be handy to put in scripts so you can make LFS installations fully automatic.
- Added explanations on the commands being executed to make it clearer why and what is being done to install the packages.
- Chapter 1: Updated the HTTP mirror list and added the FTP mirror list. This list is up-to-date as of November 14th, 2000.
- Chapter 5: In the Bash installation changed the `--with-curses` configure option to `--with-ncurses`. This seems to fix bash compilations on distribution that don't have ncurses properly installed.
- Chapter 5: Instead of having the user replace `<host>` in the gcc installation by whatever appears in `$LFS/usr/lib/gcc-lib` you can use a `*`. This won't be a problem because the `*` will expand in only one directory so the 'ln' command won't complain about it. This makes it easier to automate as well.
- Chapter 6: Mentioned the `-e` parameter to perl's Configure script that makes the script not ask you anything after it has created the `config.sh` script.
- Chapter 6: Removed the creation of the `/usr/bin/install` symlink – this symlink was already created earlier in chapter 5
- Chapter 6: Added the creation of `/var/log/lastlog` where `utmp`, `btmp` and `wtmp` are created.
- Chapter 6: When the yacc script is created in the Bison section, execute a `chmod 755` on it so we can execute the script.

Linux From Scratch

Chapter 6: Cosmetic change to the inittab file. Instead of using /dev/tty[1-6] as parameters toagetty we now use just 'dev[1-6]'. This generates a nicer output from commands like 'w'.

- Chapter 7: Modified all scripts to use absolute paths instead of relying on \$PATH to be set.
- Chapter 7: In fstab changed "none /proc proc defaults 0 0" to "proc /proc proc defaults 0 0". Upon mount problems you could get "none: device or resource busy" instead of "proc: device or resourced busy".
- Appendix C: Fixed a couple of broken links.

j.4.2 – October 11th, 2000

- Chapter 3: Newer versions were mentioned, but the links were still pointing to the older versions. Besides that I forgot to put the newer package versions in the ftp archive. Both have been fixed now.
- Chapter 5: Instead of looking at the filename of the C library files to determine which C library your starting Linux system uses we'll obtain it by running "strings /lib/libc* | grep "release version"" instead.
- Chapter 5+6: The proc file system must be mounted in chapter 5 before we enter the chroot'ed environment since after chroot the mount program will not be available yet.
- Chapter 6: Fixed a HTML bug in the GCC installation which caused a CR character to appear in certain browser.

j.4.1 – October 10th, 2000

- Removed the bash prompts from the commands. This will make it much easier to copy & paste the commands from the book onto the command line. Typing them all out is great for the first few times, but it tends to get tedious after a while. You can of course use scripts to do this all, but that's not the goal of this book. That's part of a different project (alfs.linuxfromscratch.org).
- Swapped chapters 8 and 9. Now we first reboot and then setup networking. If done the other way around, networking programs won't work unless both the normal system and the LFS system are going to run the same kernel version, which often is not the case. Swapping the chapters eliminates that possible problem.
- Chapter 3: All packages have been moved to download.linuxfromscratch.org and the links are updated accordingly. The official download sites for all the packages are listed in Appendix C.

Linux From Scratch

Chapter 5+6: Moved the execution of localedef after Glibc in chapter 5 to after you entered chroot in chapter 6. It was a mistake (the only real bug in 2.4) to put it in chapter 5.

- Chapter 6: Installing Vim as the first program. In case you need to edit something you an editor available right away. This also caused a couple of other packages to be moved to satisfy dependencies.
- Chapter 6: When we use sed to modify a Makefile file we now run make as "make -f Makefile2" instead of "mv Makefile2 Makefile && make".
- Chapter 6: Added the "publickey: files" line to the nsswitch.conf file. This is needed when you run a 2.4 kernel to login properly.
- Chapter 6: Added the /usr/bin/yacc script that runs bison with the -y switch to emulate yacc's output file name conventions. This is done because there are a few packages out there that rely on yacc and can't work with bison (yet).
- Chapter 6: Modified the /usr/sbin/makewhatis script after the installation of the man package. The /usr/sbin/makewhatis script needs the AWK= variable defined to /usr/bin/mawk.
- Chapter 7: Added the template script. This way you can easily add new bootscripts without having to write them from scratch.

j.4 – August 28th, 2000

- Split the book up into two differnet books for Intel and PPC.
- Chapter 4: Added the mail and dev/pts directories to the "Creating directories" section.
- Chapter 5: Everything from chroot and after has been put in a new chapter.
- Chapter 6: Moved the optimization part to the point just before you enter the chroot'ed environment. It's a waste to use compiler optimizations for the static packages since they will be replaced anyways.
- Chapter 6: To enter chroot we first cd to the \$LFS/root directory. Some older chroot programs have problems when you enter chroot when your starting directory isn't inside the chroot environment. Also we don't execute bash directly in the chroot'ed environment, but we start the "env" program so we can enter with a clean environment that only has CFLAGS and CXXFLAGS set.
-

Chapter 6: A few people have had problems compiling M4 in the chroot'ed environment. Instructions are provided how to install this package statically for the affected users.

- Chapter 6: We can't move the 'mv' program during the dynamic installation of the fileutils package with the mv program. So we copy it to /bin first, then remove the /usr/bin/mv one.
- Chapter 5: Added 'make localedata/install-locales' to the Glibc installation. This installs the locale files that various applications use (most notable GDK applications) if you have an NLS capable system (which LFS is, but with missing locales it's almost useless)
- Chapter 6: Moved vim's installation before Lilo since you might want to edit Lilo's Makefile file to add compiler optimization.
- Chapter 6: Moved the installation of shadow password suit after sh-utils. Else sh-utils replaces the "su" version from shadow password with it's own version which shouldn't happen.
- Chapter 6: Changed the way we enter the chroot'ed environment. We use the "env" to create an empty environment so that environment variables from the normal Linux system won't interfere in the chroot environment. The only variable set when entering the chroot'ed environment is the HOME variable.
- Chapter 6: Because of the new way we enter chroot, the \$LFS/root/.bash_profile file has been created that sets a few variables like TERM, CFLAGS, CXXFLAGS and whatever you deem necessary.

j.3.7 – August 3rd, 2000

- All chapters: Removed the <blockquote> SGML tags so that the contents of files isn't indented anymore. This improves the ease of copy and pasting from the book into your files without needing to manually reformat the files to get rid of the indentations.
- Chapter 4: Added var/tmp to the "chmod 1777 tmp usr/tmp" command.
- Chapter 4: Made mkdir commands less repetitive by putting the creation of the directories in \$LFS/usr and \$LFS/usr/local in a for-loop.
- Chapter 5: Moved the chmod 754 command for MAKEDEV after the sed operation.
- Chapter 5: Changed the order in which packages are installed to conform more to an alphabetical ordering.

Linux From Scratch

Chapter 5: After console-tools has been installed the /usr/share/defkeymap.kmap.gz file is created which will be used by the loadkeys script.

- Chapter 5: Removed "gcc -c watch.c" from "Installing Procps". Please let us know if this is still needed on certain hardware.
- Chapter 5: Added the /usr/bin/install symbolic link as it seems that at least one package (sysklogd) has the install location hard coded in it's Makefile file.
- Chapter 5: After gettext has been installed, we have a file /po-mode.el. This file will be moved to /usr/share/gettext where it probably belongs.
- Chapter 5: Instead of passing --with-root-prefix=/ to e2fsprogs' configure script, we now pass --with-root-prefix=
- Chapter 5: When gzip is installed and the files moved to /bin the hard link between the files is removed. So we just move gzip to /bin and create a symlink between gzip and gunzip.
- Chapter 5: In the chroot environment: changed the installation order of a few packages who's dependencies have changed over time.
- Chapter 5: inittab file has been slightly updated to better support the single user run level. When you change to run level S, s or 1 it will do it's job properly now.
- Chapter 6: Fixed typo in the rc script (! -f sysinit_start -> ! -f \$sysinit_start).
- Chapter 6: Changed the loadkeys command in the loadkeys script. New command is: loadkeys -d which loads the /usr/share/keymaps/defkeymap.kmap.gz file.
- Chapter 6: Changed ". /etc/init.d/functions" into "source /etc/init.d/functions".
- Chapter 6: Removed the "rm /fastboot" command from the checkfs script.

j.3.6 – July 19th, 2000

- Chapter 3: Re-ordered the software download list so it once again matches the order in which packages are used (the first package listed in the list is the first package that we will be using in the book, the second listed package will be the second package used in the book, etc).

Linux From Scratch

Chapter 3: Added the file sizes of the packages you have to download.

- Chapter 3: Removed the start–stop–daemon package.
- Chapter 3: Added the findutils and glibc patches to the package list.
- Chapter 3: Added the man–pages package to the package list.
- Chapter 4: Moved the creation of the \$LFS/dev/ files to chapter 5 after we have entered the chroot environment. This is done because GID's on normal system and LFS system might differ and the MAKEDEV script depends on the GID's.
- Chapter 5: Added the installation of the man–pages package.
- Chapter 5: Added a few commonly used groups to the /etc/group file when it is created (these are the groups needed by the MAKEDEV script).
- Chapter 5: The /proc/devices file is copied to \$LFS/proc for the benefit of the MAKEDEV script. The presence of this file ensures the proper creation of the device files.
- Chapter 5: Layout changes. Every package installation has it's own page now. Also the text from appendixa for every package is included with the installation instructions so you can read what a package is about during (or after or before) the installation of it.
- Chapter 5: Removed the patches for diffutils, grep, gzip and sed that used to fix static link problems. The problems can be fixed by passing compile arguments to the C pre–processor (cpp) instead.
- Chapter 5: Added the --disable-termcap option to configure to disable termcap backward compatibility (if you want to know why termcap isn't used anymore, please read the INSTALL file that comes with the Ncurses package).
- Chapter 5: Added a few missing files from the fileutils package to the "mv" commands.
- Chapter 5: Removed the installation of the start–stop–daemon package.
- Chapter 5: Removed the –e parameters from the make command lines.

Chapter 5: Instead of editing the procinfo, procps and psmisc Makefile files with a text editor, the sed command it used.

- Chapter 6: Added the setclock script in case your hardware clock isn't set to GMT.
- Chapter 6: Removed the use of the start-stop-daemon program and replaced them with custom functions that use programs like pidof and kill to accomplish the same tasks but with more control over what happens.
- Chapter 6: Added the loadproc and killproc functions to the /etc/init.d/functions file that take over the functions the start-stop-daemon program used to perform.
- Chapter 6: When the checkfs script runs without errors it now prints a green OK.
- Chapter 6: When /fastboot or /forcefsck exist, they won't be deleted from within the checkfs script but from within the mountfs script as soon as the root partition has been remounted in read-write mode.
- Chapter 6 & 7: Instead of sourcing a file with ". /etc/init.d/functions", "source /etc/init.d/functions" is now used. This makes it easier to read and is clearer for persons who don't know much about scripting.
- Appendix A: removed start-stop-daemon.
- Appendix B: Removed a few unrelated items from the book and howto sections (the references to Sendmail and ISP-Hookup-HOWTO).

j.3.5 – June 19th, 2000

- Chapter 3: Updated LILO download location
- Chapter 3: Updated Shadow Password Suite download location
- Chapter 3: Updated the Flex download location
- Chapter 3: Updated the File download location
- Chapter 3: Added netkit-base and net-tools to the mandatory packages section

Linux From Scratch

Chapter 5: A glibc-2.1.3 patch is available if you have problems compiling glibc on a bash-2.04 machine.

- Chapter 5: Added compiler optimization
- Chapter 5: Added the creation of the root password to "Configuring essential software"
- Chapter 5: The Linux86 package has been replaced by the Bin86 package.
- Chapter 5: Included information on how to optimize compilations.
- Chapter 5: Moved installation of Groff and Man before Perl. This way Perl known how to install man pages and where to install them.
- Chapter 5: Changed GCC's local-prefix option to /usr/local instead of /usr (this was still a residue from the time where /usr/local was a symbolic link to /usr)
- Chapter 5: Fixed the commands when a patch is used and the patch filename contained the .gz suffix.
- Chapter 5: Added --disable-nls to every configure command in the "Perparing the LFS system..." section which didn't have it yet.
- Chapter 5: Added the installation of bash-2.03 so you have a shell that can be used to compile packages that violate POSIX standards regarding valid characters in variable names
- Chapter 5: Added the installation of console-tools and console-data for people who have non-US keyboards
- Chapter 5: Moved the ed program to the /bin directory conforming the FHS standard
- Chapter 6 & 7: Implemented LSB recommended run level scheme.
- Chapter 6 & 7: Implemented "fancy bootscripts". When something fails in a bootscript it still says FAILED but the text red. When something succeeded it still will print OK but the text is green.
- Chapter 6: Added the loadkeys scripts for people with non-US keyboards

Linux From Scratch

Chapter 6: Added the `/etc/sysconfig` directory to "Creating directories"

- Chapter 6: Renamed the `checkroot` boot script into `checkfs`. The script also checks other file systems now.
- Chapter 6: Updated the `mountfs` boot script to mount all file systems that are mentioned in the `/etc/fstab` file and don't have the `noauto` option set.
- Chapter 6: After `checkfs` evaluated the existence of `/fastboot` or `/forcecheck` it will remove those files.
- Chapter 6 & 7: Changed the mode of the boot scripts from 755 to 754
- Chapter 7: Moved system specific information for hostname and ethernet configuration to the `/etc/sysconfig/network` file
- Chapter 7: Removed the default gateway command
- Chapter 7: Fixed the typo in the `ethnet` script (`NETMAKSK` → `NETMASK`)
- Chapter 7: A `net-tools` patch is available to fix a minor bug in the package (illegal variable names that `bash-2.04` will complain about)

j.3.4 – June 5th, 2000

- Chapter 5: Fixed the kernel header files configuration
- Chapter 5: Fixed the `lilo` configuration

j.3.3 – May 15th, 2000

- Changed the default mount point from `/mnt/xxx` to `/mnt/lfs` (where `xxx` used to be the partition's designation like `hda5`, `sda5` and others). The reason for the change is to make cross-platform instructions easier.
- Chapter 4: Changed the default modes for the `$LFS/root` and `$LFS/tmp` directory to respectively 0750 and 1777.
-

Chapter 5: Removed the encoded password from the passwd file. Instead a file with no set password is created. The root password can be set by the user when the system is rebooted into the LFS system (after chapter 8).

- Chapter 5: Fixed the procps compile command for watch.c. It should compile properly now.
- Chapter 5: Fixed gzip patch installation (used the wrong filename in the patch command)
- Chapter 5: Changed 'entering the chroot'ed environment' to make bash a login shell.
- Chapter 5: Configuring the kernel has been moved to this chapter because it needs to be done before programs like e2fsprogs and lilo are compiled.
- Chapter 6: Fixed the rc script. It now checks to see if the previous run level starts a service before attempting to stop it in the new run level. Also, if a service is already started in the previous run level it won't attempt to start the service in the new run level again. Thanks to Jason Pearce for providing this fixed script.
- Chapter 7: Fixed the ethernet script – removed parentheses from the environment variables and removed the command to add a route. The ifconfig command used to bring the eth device up already sets this route.

j.3.2 – April 18th, 2000

- Chapter 4.7: Change only the owner of the \$LFS/dev/* files
- Fixed a large amount of typo's that occurred during the transition from the LinuxDoc DTD (2.2 and lower) to the DocBook DTD (2.3.1 and higher).
- Moved chapters around quite a bit and applied a new structure in the book. Installations for Intel, Apple PowerPC and future systems will be put in their own dedicated part of the book.
- After the system is prepared to install the basic system software, we no longer reboot the system but instead we setup a chroot'ed environment. This will have the same effect without having to reboot.
- Apple PowerPC has its own dedicated chapters now. This should increase readability a lot
- All optional chapters have been removed. LFS follows a "we provide the foundation, it's up to you to build the rest of the house" philosophy.

Replaced the fixed packages by patch files. This way you can see what needs to be changed in a package in order to get it to compile properly.

j.3.1 – April 12th, 2000

- Chapter 4.4: Added the `$LFS/usr/info` symlink which points to `$LFS/usr/share/info`
 - Chapter 7.3.1: Added a second variation to a 'swap-line' in a `fstab` file.
 - Chapter 7.3.2: Removed `$LFS` from the commands.
 - Chapter 7.4.43: Added the `vi` symlink
 - Chapter 9.2.5: Improved `ethnet` script to include routing information
 - Chapter 10.1.2: Fixed missing subdirectory 'mqueue' in `mkdir /var/spool -> /mkdir /var/spool/mqueue`
 - Chapter 10.1.4: Updated the `sendmail` configuration file with a few necessary options
 - Chapter 10.1.7: Fixed wrong directory path `/etc/init.d/rc2.d -> /etc/rc2.d`
-

Mailinglists and archives

The linuxfromscratch.org server is hosting the following public accessible mailinglists:

- lfs–discuss
 - lfs–apps
 - lfs–announce
 - alfs–discuss
 - alfs–docs
 - alfs–ipc
 - alfs–profile
-

lfs–discuss

The lfs–discuss mailinglist discusses matters strictly related to the LFS–BOOK. If you have problems with the book, want to report a bug or two or have suggestions to improve the book, use this mailinglist.

Any other mail is to be posted on the lfs–apps list.

lfs–apps

The lfs–apps list deals with everything that does not fit on the lfs–discuss list.

lfs–announce

The lfs–announce list is a moderated list. You can subscribe to it, but you can't post any messages to this list. This list is used to announce new stable releases. If you want to be informed about development releases as well then you'll have to join the lfs–discuss list. If you're already on the lfs–discuss list there's little use subscribing to this list as well because everything that is posted to the lfs–announce list will be posted to the lfs–discuss list as well.

alfs–discuss

The alfs–discuss list discusses the development of ALFS which stands for Automated LinuxFromScratch. The goal of this project is to develop an installation tool that can install an LFS system automatically for you. It's main goal is to speed up compilation by taking away your need to manually enter the commands to configure, compile and install packages.

alfs–docs

ALFS–docs is the ALFS documentation project which creates and maintains all of the ALFS documentation.

alfs–ipc

The alfs–ipc list discusses the ALFS InterProcess Communication issues.

alfs–profile

The alfs–profile list discusses the development of the ALFS XML profile and DTD.

How to subscribe?

You can subscribe to any of the above mentioned mailinglists by sending an email to listar@linuxfromscratch.org and write *subscribe listname* in the subject line of the message, where listname is replaced by either lfs–discuss, lfs–config, lfs–apps, lfs–announce, linux, alfs–discuss, alfs–profile, alfs–pr, alfs–docs, alfs–ipc or alfs–license

You can, if you want, subscribe to multiple lists at the same time using one email. If you want to do so, write some junk in the subject line, something that isn't a valid command like "hello". Then write the subscribe commands in the body of the message. The email will look like:

```
To: listar@linuxfromscratch.org
Subject: hello
```

```
subscribe lfs–discuss
subscribe lfs–apps
subscribe alfs–discuss
```

After you have sent the email, the Listar program will send you an email back requesting a confirmation of

your subscription request. After you have sent back this confirmation email, Majordomo will send you an email again with the message that you have been subscribed to the list(s) along with an introduction message for that particular list.

How to unsubscribe?

To unsubscribe from a list, send an email to listar@linuxfromscratch.org and write *unsubscribe listname* in the subject line of the message, where listname is replaced by either lfs–discuss, lfs–config, lfs–apps, lfs–announce, linux, alfs–discuss, alfs–profile, alfs–pr, alfs–docs, alfs–ipc or alfs–license

You can, if you want, unsubscribe from multiple lists at the same time using one email. If you want to do so, write some junk in the subject line, something that isn't a valid command like "hello". Then write the unsubscribe commands in the body of the message. The email will look like:

```
To: listar@linuxfromscratch.org
Subject: hello
```

```
unsubscribe lfs–discuss
unsubscribe lfs–apps
unsubscribe alfs–discuss
```

After you have sent the email, the Listar program will send you an email back requesting a confirmation of your unsubscription request. After you have sent back this confirmation email, Listar will send you an email again with the message that you have been unsubscribed from the list(s).

Mail archives

Every publically available mailinglist has a mailinglist archive you can access to find information on subjects already posted to this list. You can find them at <http://archive.linuxfromscratch.org/mail-archives>.

Contact information

Direct all your emails to the [lfs-discuss](#) mailinglist preferably.

If you need to reach Gerard Beekmans personally, send an email to gerard@linuxfromscratch.org

If you need to reach Michael Peters, Apple PPC maintainer of this book, personally, send an email to mpters@mac.com

Chapter 2. Important information

About \$LFS

Please read the following carefully: throughout this book you will frequently see the variable name \$LFS. \$LFS must at all times be replaced by the directory where the partition that contains the LFS system is mounted. How to create and where to mount the partition will be explained in full detail later on in chapter 4. In my case the LFS partition is mounted on /mnt/lfs. If I read this book myself and I see \$LFS somewhere, I will pretend that I read /mnt/lfs. If I read that I have to run this command: `cp inittab $LFS/etc` I actually will run this: `cp inittab /mnt/lfs/etc`

It's important that you do this no matter where you read it; be it in commands you enter on the prompt, or in a file you edit or create.

If you want, you can set the environment variable LFS. This way you can literally enter \$LFS instead of replacing it by something like /mnt/lfs. This is accomplished by running: `export LFS=/mnt/lfs`

If I read `cp inittab $LFS/etc`, I literally can type `cp inittab $LFS/etc` and the shell will replace this command by `cp inittab /mnt/lfs/etc` automatically.

Do not forget to set the \$LFS variable at all times. If you haven't set the variable and you use it in a command, \$LFS will be ignored and whatever is left will be executed. The command `cp inittab $LFS/etc` without the \$LFS variable set, will result in copying the inittab file to the /etc directory which will overwrite your system's inittab. A file like inittab isn't that big a problem as it can easily be restored, but if you would make this mistake during the installation of the C Library, you can damage things.

How to download the software

Throughout this document I will assume that you have stored all the packages you have downloaded somewhere in `$LFS/usr/src`.

I use the convention of having a `$LFS/usr/src/sources` directory. Under sources you'll find the directory 0–9 and the directories a through z. A package as `sysvinit–2.78.tar.gz` is stored under `$LFS/usr/src/sources/s/` A package as `bash–2.04.tar.gz` is stored under `$LFS/usr/src/sources/b/` and so forth. You don't have to follow this convention of course, I was just giving an example. It's better to keep the packages out of `$LFS/usr/src` and move them to a subdirectory, so we'll have a clean `$LFS/usr/src` directory in which we will unpack the packages and work with them.

The next chapter contains the list of all the packages you need to download, but the partition that is going to contain our LFS system isn't created yet. Therefore store the files temporarily somewhere where you want and remember to copy them to `$LFS/usr/src/` when you have finished the chapter in which you prepare a new partition (chapter 4).

How to install the software

Before you can actually start doing something with a package, you need to unpack it first. Often you will find the package files being tar'ed and gzip'ed (you can determine this by looking at the extension of the file. tar'ed and gzip'ed archives have a .tar.gz or .tgz extension for example)). I'm not going to write down every time how to ungzp and how to untar an archive. I will tell you how to do that once, in this paragraph. There is also the possibility that you have the ability of downloading a .tar.bz2 file. Such a file is tar'ed and compressed with the bzip2 program. Bzip2 achieves a better compression than the commonly used gzip does. In order to use bz2 archives you need to have the bzip2 program installed. Most if not every distribution comes with this program so chances are high it is already installed on your system. If not, install it using your distribution's installation tool.

To start with, change to the \$LFS/usr/src directory by running:

```
cd $LFS/usr/src
```

When you have a file that is tar'ed and gzip'ed, you unpack it by running either one of the following two commands, depending on the filename format:

```
tar xvzf filename.tar.gz
tar xvzf filename.tgz
```

When you have a file that is tar'ed and bzip'ed, you unpack it by running:

```
bzcat filename.tar.bz2 | tar xv
```

Some tar programs (most of them nowadays but not all of them) are slightly modified to be able to use bzip2 files directly using either the I or the y tar parameter which works the same as the z tar parameter to handle gzip archives.

When you have a file that is tar'ed, you unpack it by running:

```
tar xvf filename.tar
```

When the archive is unpacked a new directory will be created under the current directory (and this document assumes that you unpack the archives under the `$LFS/usr/src` directory). You have to enter that new directory before you continue with the installation instructions. So everytime the book is going to install a program, it's up to you to unpack the source archive.

After you have installed a package you can do two things with it. You can either delete the directory that contains the sources or you can keep it. If you decide to keep it, that's fine by me. But if you need the same package again in a later chapter you need to delete the directory first before using it again. If you don't do this, you might end up in trouble because old settings will be used (settings that apply to your normal Linux system but which don't always apply to your LFS system). Doing a simple `make clean` or `make distclean` does not always guarantee a totally clean source tree. The configure script can also have files lying around in various subdirectories which aren't always removed by a `make clean` process.

There is on exception to that rule: don't remove the linux kernel source tree. A lot of programs need the kernel headers, so that's the only directory you don't want to remove, unless you are not going to compile any software anymore.

II. Part II – Installing the LFS system

Table of Contents

3. [Packages you need to download](#)
 4. [Preparing a new partition](#)
 5. [Preparing the LFS system](#)
 6. [Installing basic system software](#)
 7. [Creating system boot scripts](#)
 8. [Making the LFS system bootable](#)
 9. [Setting up basic networking](#)
-

Chapter 3. Packages you need to download

Below is a list of all the packages you need to download for building the basic system. The version numbers printed correspond to versions of the software that is known to work and which this book is based on. If you experience problems which you can't solve yourself, download the version that is assumed in this book (in case you downloaded a newer version).

If the packages.linuxfromscratch.org server isn't allowing connections anymore try one of our mirror sites. The addresses of the mirror sites can be found in [Chapter 1 – Book Versions](#)

We have provided a list of official download sites of the packages below in Appendix C. The LFS FTP archive only contains the versions of packages that are recommended for use in this book. If you're looking for newer versions than the ones listed here (by the time you read this, a couple of new versions will already be available, especially non-GNU software) you can have a look in Appendix C.

Please note that all files downloaded from the LFS FTP archive are files compressed with bzip2 instead of gz. If you don't know how to handle bz2 files, please read [Chapter 2 – How to install the software](#).

- Browse FTP: <ftp://packages.linuxfromscratch.org>
- Bash (2.04) – 1,307 KB:
<ftp://packages.linuxfromscratch.org/pub/common-packages/bash-2.04.tar.bz2>
- Binutils (2.10) 5,430 KB:
<ftp://packages.linuxfromscratch.org/pub/common-packages/binutils-2.10.tar.bz2>
- Bzip2 (1.0.1) 454 KB:
<ftp://packages.linuxfromscratch.org/pub/common-packages/bzip2-1.0.1.tar.gz>
- Diff Utils (2.7) 247 KB:
<ftp://packages.linuxfromscratch.org/pub/common-packages/diffutils-2.7.tar.bz2>
- File Utils (4.0) 801 KB:
<ftp://packages.linuxfromscratch.org/pub/common-packages/fileutils-4.0.tar.bz2>
- GCC (2.95.2) 9,555 KB:
<ftp://packages.linuxfromscratch.org/pub/common-packages/gcc-2.95.2.tar.bz2>
- Linux Kernel (2.2.17) 13,727 KB:
<ftp://packages.linuxfromscratch.org/pub/common-packages/linux-2.2.17.tar.bz2>
-

Glibc (2.1.3) 6,308 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/glibc-2.1.3.tar.bz2>

•

Glibc-crypt (2.1) 36 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/glibc-crypt-2.1.tar.bz2>

•

Glibc-linuxthreads (2.1.3) 112 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/glibc-linuxthreads-2.1.3.tar.bz2>

•

Glibc Patch (2.1.3) – 1 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/glibc-2.1.3.patch.gz>

•

Grep (2.4.2) 382 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/grep-2.4.2.tar.bz2>

•

Gzip (1.2.4a) 178 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/gzip-1.2.4a.tar.bz2>

•

Gzip Patch (1.2.4a) 1 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/gzip-1.2.4a.patch.gz>

•

Make (3.79.1) 749 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/make-3.79.1.tar.bz2>

•

Sed (3.02) 221 KB: <ftp://packages.linuxfromscratch.org/pub/common-packages/sed-3.02.tar.bz2>

•

Sh-utils (2.0) 824 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/sh-utils-2.0.tar.bz2>

•

Tar (1.13) 730 KB: <ftp://packages.linuxfromscratch.org/pub/common-packages/tar-1.13.tar.bz2>

•

Text Utils (2.0) 1,040 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/textutils-2.0.tar.bz2>

•

MAKEDEV (2.5) – 11 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/MAKEDEV-2.5.tar.bz2>

•

Bison (1.28) 321 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/bison-1.28.tar.bz2>

•

Mawk (1.3.3) 168 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/mawk1.3.3.tar.bz2>

•

Patch (2.5.4) 149 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/patch-2.5.4.tar.bz2>

•

Find Utils (4.1) 226 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/findutils-4.1.tar.bz2>

•

Find Utils Patch (4.1) 1 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/findutils-4.1.patch.gz>

•

Ncurses (5.1) 1,280 KB:

<ftp://packages.linuxfromscratch.org/pub/intel-packages/ncurses-5.1.tar.bz2>

•

Less (358) 178 KB: <ftp://packages.linuxfromscratch.org/pub/common-packages/less-358.tar.bz2>

•

Groff (1.16.1) 1,173 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/groff-1.16.1.tar.bz2>

•

Man (1.5h1) 156 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/man-1.5h1.tar.bz2>

•

Perl (5.6.0) 4,327 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/perl-5.6.0.tar.bz2>

•

M4 (1.4) 249 KB: <ftp://packages.linuxfromscratch.org/pub/common-packages/m4-1.4.tar.bz2>

•

Texinfo (4.0) 812 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/texinfo-4.0.tar.bz2>

•

Autoconf (2.13) 333 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/autoconf-2.13.tar.bz2>

•

Automake (1.4) 277 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/automake-1.4.tar.bz2>

•

Flex (2.5.4a) 278 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/flex-2.5.4a.tar.bz2>

•

File (3.32) 125 KB: <ftp://packages.linuxfromscratch.org/pub/common-packages/file-3.32.tar.bz2>

•

Libtool (1.3.5) 361 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/libtool-1.3.5.tar.bz2>

•

Bin86 (0.15.1) 112 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/bin86-0.15.1.tar.bz2>

•

Gettext (0.10.35) 525 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/gettext-0.10.35.tar.bz2>

•

Console-tools (0.2.3) 490 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/console-tools-0.2.3.tar.bz2>

•

Console-tools (0.2.3) Patch: 4 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/console-tools-0.2.3.patch.gz>

•

Console-data (1999.08.29) 418 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/console-data-1999.08.29.tar.bz2>

•

E2fsprogs (1.19) 808 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/e2fsprogs-1.19.tar.bz2>

•

Ed (0.2) 158 KB: <ftp://packages.linuxfromscratch.org/pub/common-packages/ed-0.2.tar.bz2>

•

Ld.so (1.9.9) 280 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/ld.so-1.9.9.tar.bz2>

•

Lilo (21.6) 172 KB: <ftp://packages.linuxfromscratch.org/pub/intel-packages/lilo-21.6.tar.bz2>

•

Modutils (2.3.17) 172 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/modutils-2.3.17.tar.bz2>

Vim-rt (5.7) 905 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/vim-5.7-rt.tar.bz2>

•

Vim-src (5.7) 963 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/vim-5.7-src.tar.bz2>

•

Procinfo (17) 21 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/procinfo-17.tar.bz2>

•

Procps (2.0.7) 153 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/procps-2.0.7.tar.bz2>

•

Psmisc (19) 20 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/psmisc-19.tar.bz2>

•

Shadow Password Suite (20000902) 557 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/shadow-20000902.tar.bz2>

•

Sysklogd (1.4) 67 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/sysklogd-1.4.tar.bz2>

•

Sysvinit (2.78) 90 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/sysvinit-2.78.tar.bz2>

•

Util Linux (2.10o) 871 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/util-linux-2.10o.tar.bz2>

•

Man-pages (1.31) 468 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/man-pages-1.31.tar.bz2>

•

Netkit-base (0.17) 49 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/netkit-base-0.17.tar.bz2>

•

Net-tools (1.57) 187 KB:

<ftp://packages.linuxfromscratch.org/pub/common-packages/net-tools-1.57.tar.bz2>

•

Total size of all intel-packages: 60,017 KB (58,61 MB)

Chapter 4. Preparing a new partition

Introduction

In this chapter the partition that is going to host the LFS system is going to be prepared. A new partition will be created, an ext2 file system will be created on it and the directory structure will be created. When this is done, we can move on to the next chapter and start building a new Linux system from scratch.

Creating a new partition

Before we can build our new Linux system, we need to have an empty Linux partition on which we can build our new system. I recommend a partition size of around 750 MB. This gives you enough space to store all the tarballs and to compile all packages without worrying running out of the necessary temporary disk space. If you already have a Linux Native partition available, you can skip this subsection.

Start the fdisk program (or another fdisk program you prefer) with the appropriate hard disk as the option (like /dev/hda if you want to create a new partition on the primary master IDE disk). Create a Linux Native partition, write the partition table and exit the fdisk program. If you get the message that you need to reboot your system to ensure that the partition table is updated, then please reboot your system now before continuing. Remember what your new partition's designation is. It could be something like hda11 (as it is in my case). This newly created partition will be referred to as the LFS partition in this book.

Creating a ext2 file system on the new partition

Once the partition is created, we have to create a new ext2 file system on that partition. To create a new ext2 file system we use the `mke2fs` command. Enter the new partition as the only option and the file system will be created. If your partition is `hda11`, you would run:

```
mke2fs /dev/hda11
```

Mounting the new partition

Now that we have created the ext2 file system, it is ready for use. All we have to do to be able to access it (as in reading from and writing data to it) is mounting it. If you mount it under `/mnt/lfs`, you can access this partition by going to the `/mnt/lfs` directory and then do whatever you need to do. This book will assume that you have mounted the partition on a subdirectory under `/mnt`. It doesn't matter which directory you choose, just make sure you remember what you chose.

Create the `/mnt/lfs` directory by running:

```
mkdir -p /mnt/lfs
```

Now mount the LFS partition by running:

```
mount /dev/xxx /mnt/lfs
```

Replace "xxx" by your partition's designation.

This directory (`/mnt/lfs`) is the `$LFS` variable you have read about earlier. So if you read somewhere to "cp inittab `$LFS/etc`" you actually will type "cp inittab `/mnt/lfs/etc`". Or if you want to use the `$LFS` environment variable, execute **export LFS=/mnt/lfs** now.

Creating directories

Let's create the directory tree on the LFS partition according to the FHS standard which can be found at <http://www.pathname.com/fhs/>. Issuing the following commands will create the necessary directories:

```
cd $LFS
mkdir bin boot dev dev/pts etc home lib mnt proc root sbin
tmp var
for dirname in $LFS/usr $LFS/usr/local
do
  mkdir $dirname
  cd $dirname
  mkdir bin etc include lib sbin share src tmp var
  ln -s share/man man
  ln -s share/doc doc
  ln -s share/info info
  cd $dirname/share
  mkdir dict doc info locale man nls misc terminfo zoneinfo
  cd $dirname/share/man
  mkdir man1 man2 man3 man4 man5 man6 man7 man8
done
cd $LFS/var
mkdir lock log mail run spool tmp
```

Normally directories are created with permission mode 755, which isn't desired for all directories. I haven't checked the FHS if they suggest default modes for certain directories, so I'll just change the modes for a few directories that make sense to change. The first change is a mode 0750 for the \$LFS/root directory. This is to make sure that not just everybody can enter the /root directory (the same you would do with /home/username directories). The second change is a mode 1777 for the tmp directories. This way every user can write stuff to the /tmp directory if they need to. The sticky (1) bit makes sure users can't delete other user's file which they normally can do because the directory is set in such a way that every body (owner, group, world) can write to that directory.

```
cd $LFS &&
chmod 0750 root &&
chmod 1777 tmp usr/tmp var/tmp
```

Now that the directories are created, copy the source files you have downloaded in chapter 3 to some subdirectory under \$LFS/usr/src (you will need to create this subdirectory yourself).

Chapter 5. Preparing the LFS system

Introduction

In the following chapters we will install all the software that belongs to a basic Linux system. After you're done with this chapter you have a fully working Linux system. The remaining chapters deal with setting up networking, creating the boot scripts and adding an entry to `lilo.conf` so that you can boot your LFS system.

The software in this chapter will be linked statically. These programs will be re-installed in the next chapter and linked dynamically. The reason for the static version first is that there is a chance that our normal Linux system and your LFS system aren't using the same C Library versions. If the programs in the first part are linked against an older C library version, those programs might not work well on the LFS system.

The key to learn what makes Linux tick is to know exactly what packages are used for and why you or the system needs them. Descriptions of the package content are provided after the Installation subsection of each package and in Appendix A as well.

We're about to start with installing the first set of packages. These packages will be, as previously explained, linked statically.

During the installation of various packages you will most likely see compiler warnings scrolling by on your screen. These are normal and can be safely ignored. They are just that, warnings (mostly about improper use of the C or C++ syntax, but not illegal use. It's just that often C standards changed and packages still use the old standard which is not a problem).

Before we start, make sure you have the LFS environment variable setup if you plan on using it, by running the following command:

```
echo $LFS
```

Installing Bash

Installation of Bash

Install Bash by running the following commands:

```
./configure --enable-static-link --prefix=$LFS/usr \  
  --bindir=$LFS/bin --disable-nls --with-ncurses &&  
make &&  
make install &&  
cd $LFS/bin &&  
ln -s bash sh
```

If you get errors when compiling bash that tell you about not being able to find "–lcurse" run these two commands to create the missing symlink (so far we have not encountered one distribution that has this libncurses symlink setup properly, except for LFS systems where it is setup properly):

```
cd /usr/lib &&  
ln -s libncurses.a libcurse.a
```

Note: Normally the libncurses.a file resides in the /usr/lib directory but it might reside in /lib (like it does on LFS systems). So check to make sure whether you should run the ln command in /usr/lib or in /lib

Command explanations

--enable-static-link: This configure option causes Bash to be linked statically

--prefix=\$LFS/usr: This configure option installs all of Bash's files under the \$LFS/usr directory, which becomes the /usr directory after you chroot into \$LFS or when you reboot the system into LFS.

--bindir=\$LFS/bin: This installs the executable files in \$LFS/bin. We do this because we want bash to be in /bin, not in /usr/bin. One reason being: your /usr partition might be on a separate partition which has to be mounted at some point. Before that partition is mounted you need and will want to have bash available (it will be hard to execute the boot scripts without a shell for instance).

--disable-nls: This disables the build of NLS (National Language Support). It's only a waste of time for now as Bash will be reinstalled in the next chapter.

--with-ncurses: This causes Bash to be linked against the ncurses library instead of the default termcap library which is becoming obsolete.

ln -s bash sh: This command creates the sh symlink that points to bash. Most scripts run themselves via 'sh'; sh being a symlink to the default system shell. Because programs and scripts don't know what shell you use by default (could be bash, ksh, korn, tch, csh and others) they use the common symlink sh which, if the system is properly setup, always points to the system's default shell.

The **&&**'s at the end of every line cause the next command only to be executed when the previous command exists with a return value of 0 indicating success. In case you copy&paste all of these commands on the shell you want to be ensured that if ./configure fails, make isn't being executed and likewise if make fails that make install isn't being executed, and so forth.

Contents

The Bash package contains the bash program

Description

Bash is the Bourne–Again SHell, which is a widely used command interpreter on Unix systems. Bash is a program that reads from standard input, the keyboard. You type something and the program will evaluate what you have typed and do something with it, like running a program.

Installing Binutils

Installation of Binutils

Install Binutils by running the following commands:

```
./configure --prefix=$LFS/usr --disable-nls &&  
make -e LDFLAGS=-all-static tooldir=$LFS/usr &&  
make -e tooldir=$LFS/usr install
```

Command explanations

make -e: The `-e` parameter tells make that environment variables take precedence over variables defined in the Makefile file(s). This is needed in order to successfully link binutils statically.

LDFLAGS=-all-static: Setting the variable LDFLAGS to the value `-all-static` causes binutils to be linked statically.

tooldir=\$LFS/usr: Normally the `tooldir` (the directory where the executables from binutils end up in) is set to `$(exec_prefix)/$(target_alias)` which expands into, for example, `/usr/i686-pc-linux-gnu`. Since we only build for our own system we don't need this target specific directory in `$LFS/usr`. You would use that setup if you use your system to cross-compile (for example you would compile a package on your Intel machine that generates code that can be executed on Apple PowerPC machines).

Description

The Binutils package contains the `ld`, `as`, `ar`, `nm`, `objcopy`, `objdump`, `ranlib`, `size`, `strings`, `strip`, `c++filt`, `addr2line` and `nlmconv` programs

Description

ld

`ld` combines a number of object and archive files, relocates their data and ties up symbol references. Often the last step in building a new compiled program to run is a call to `ld`.

as

as is primarily intended to assemble the output of the GNU C compiler gcc for use by the linker ld.

ar

The ar program creates, modifies, and extracts from archives. An archive is a single file holding a collection of other files in a structure that makes it possible to retrieve the original individual files (called members of the archive).

nm

nm lists the symbols from object files.

objcopy

objcopy utility copies the contents of an object file to another. objcopy uses the GNU BFD Library to read and write the object files. It can write the destination object file in a format different from that of the source object file.

objdump

objdump displays information about one or more object files. The options control what particular information to display. This information is mostly useful to programmers who are working on the compilation tools, as opposed to programmers who just want their program to compile and work.

ranlib

ranlib generates an index to the contents of an archive, and stores it in the archive. The index lists each symbol defined by a member of an archive that is a relocatable object file.

size

size lists the section sizes —and the total size— for each of the object files objfile in its argument list. By default, one line of output is generated for each object file or each module in an archive.

strings

For each file given, strings prints the printable character sequences that are at least 4 characters long (or the number specified with an option to the program) and are followed by an unprintable character. By default, it

only prints the strings from the initialized and loaded sections of object files; for other types of files, it prints the strings from the whole file.

strings is mainly useful for determining the contents of non-text files.

strip

strip discards all or specific symbols from object files. The list of object files may include archives. At least one object file must be given. strip modifies the files named in its argument, rather than writing modified copies under different names.

c++filt

The C++ language provides function overloading, which means that you can write many functions with the same name (providing each takes parameters of different types). All C++ function names are encoded into a low-level assembly label (this process is known as mangling). The c++filt program does the inverse mapping: it decodes (demangles) low-level names into user-level names so that the linker can keep these overloaded functions from clashing.

addr2line

addr2line translates program addresses into file names and line numbers. Given an address and an executable, it uses the debugging information in the executable to figure out which file name and line number are associated with a given address.

nlmconv

nlmconv converts relocatable object files into the NetWare Loadable Module files, optionally reading header files for NLM header information.

Installing Bzip2

Installation of Bzip2

Install Bzip2 by running the following commands:

```
sed \
  s/"\$(CC) \$(CFLAGS) -o"/"\$(CC) \$(CFLAGS) \$(LDFLAGS)
-o"/ \
  Makefile | make -f - LDFLAGS=-static &&
make PREFIX=$LFS/usr install &&
cd $LFS/usr/bin &&
mv bzip2 bunzip2 bzip2recover $LFS/bin
```

Command explanations

sed: The sed command here searches for the string "\$\$(CC) \$(CFLAGS) -o" and replaced it by "\$\$(CC) \$(CFLAGS) \$(LDFLAGS) -o" in the Makefile file. We make that modification so it will be easier to link bzip2 statically.

...Makefile | make -f -: Makefile is the last parameter of the sed command which indicates the file to search and replace in. sed normally sends the modified file to stdout (standard output) which will be your console. With the construction we use, sed's output will be piped to the make program. Normally when make is started it tries to find a number of files like Makefile. But we have modified the Makefile file so we don't want make to use it. The "-f -" parameter tells make to read it's input from another file, or from stdin (standard input) which the dash (-) implies. This is one way to do it. Another way would be to have sed write the output to a different file and tell make with the -f parameter to read that alternate file.

LDFLAGS=-static: This is the second way we use to link a package statically. This is also the most common way. As you'll notice, the -all-static value is only used with the binutils package and won't be used throughout the rest of this book.

Contents

The Bzip2 packages contains the bzip2, bunzip2, bzip2recover programs.

Description

Bzip2

bzip2 compresses files using the Burrows–Wheeler block sorting text compression algorithm, and Huffman coding. Compression is generally considerably better than that achieved by more conventional LZ77/LZ78–based compressors, and approaches the performance of the PPM family of statistical compressors.

Bunzip2

Bunzip2 decompresses files that are compressed with bzip2.

bzcat

bzcat (or bzip2 -dc) decompresses all specified files to the standard output.

bzip2recover

bzip2recover recovers data from damaged bzip2 files.

Installing Diffutils

Installation of Diffutils

Install Diffutils by running the following commands:

```
CPPFLAGS=-Dre_max_failures=re_max_failures2 \  
./configure --prefix=$LFS/usr --disable-nls &&  
make LDFLAGS=-static &&  
make install
```

Command explanations

CPPFLAGS=-Dre_max_failures=re_max_failures2: The CPPFLAGS variable is a variable that's read by the cpp program (C PreProcessor). The value of this variable tells the preprocessor to replace every instance of re_max_failures it finds by re_max_failures2 before handing the source file to the compiler itself for compilation. This package has problems linking statically on certain platforms (depending on the Glibc version used on that system) and this construction fixes that problem.

Contents

The Diffutils package contains the cmp, diff, diff3 and sdiff programs.

Description

cmp and diff

cmp and diff both compare two files and report their differences. Both programs have extra options which compare files in different situations.

diff3

The difference between diff and diff3 is that diff compares 2 files, diff3 compares 3 files.

sdiff

sdiff merges two two files and interactively outputs the results.

Installing Fileutils

Installation of Fileutils

Install Fileutils by running the following commands:

```
./configure --disable-nls --prefix=$LFS/usr --bindir=$LFS/bin
&&
make LDFLAGS=-static &&
make install &&
cd $LFS/usr/bin &&
ln -s ../../bin/install install
```

Contents

The Fileutils package contains the chgrp, chmod, chown, cp, dd, df, dir, dircolors, du, install, ln, ls, mkdir, mkfifo, mknod, mv, rm, rmdir, sync, touch and vdir programs.

Description

chgrp

chgrp changes the group ownership of each given file to the named group, which can be either a group name or a numeric group ID.

chmod

chmod changes the permissions of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new permissions.

chown

chown changes the user and/or group ownership of each given file.

cp

cp copies files from one place to another.

dd

dd copies a file (from the standard input to the standard output, by default) with a user-selectable blocksize, while optionally performing conversions on it.

df

df displays the amount of disk space available on the filesystem containing each file name argument. If no file name is given, the space available on all currently mounted filesystems is shown.

ls, dir and vdir

dir and vdir are versions of ls with different default output formats. These programs list each given file or directory name. Directory contents are sorted alphabetically. For ls, files are by default listed in columns, sorted vertically, if the standard output is a terminal; otherwise they are listed one per line. For dir, files are by default listed in columns, sorted vertically. For vdir, files are by default listed in long format.

dircolors

dircolors outputs commands to set the LS_COLOR environment variable. The LS_COLOR variable is used to change the default color scheme used by ls and related utilities.

du

du displays the amount of disk space used by each argument and for each subdirectory of directory arguments.

install

install copies files and sets their permission modes and, if possible, their owner and group.

ln

ln makes hard or soft (symbolic) links between files.

mkdir

mkdir creates directories with a given name.

mkfifo

mkfifo creates a FIFO with each given name.

mknod

mknod creates a FIFO, character special file, or block special file with the given file name.

mv

mv moves files from one directory to another or renames files, depending on the arguments given to mv.

rm

rm removes files or directories.

rmdir

rmdir removes directories, if they are empty.

sync

sync forces changed blocks to disk and updates the super block.

touch

touch changes the access and modification times of each given file to the current time. Files that do not exist are created empty.

Installing GCC on the normal system if necessary

Installation of GCC on the normal system if necessary

In order to compile Glibc-2.1.3 later on you need to have gcc-2.95.2 installed. Although any GCC version above 2.8 would do, 2.95.2 is the highly recommended version to use. egcs-2.91.x is also known to work. If you don't have gcc-2.95.x or egcs-2.91.x you need to install gcc-2.95.2 on your normal system before you can compile Glibc later in this chapter.

To find out which compiler version your system has, run the following command:

```
gcc --version
```

If your normal Linux system does not have gcc-2.95.x or egcs-2.91.x installed you need to install it now. We won't replace the current compiler on your system, but instead we will install gcc in a separate directory (/usr/local/gcc2952). This way no binaries or header files will be replaced.

After you unpacked the gcc-2.95.2 archive don't enter the newly created gcc-2.95.2 directory but stay in the \$LFS/usr/src directory. Install GCC by running the following commands:

```
mkdir $LFS/usr/src/gcc-build &&
cd $LFS/usr/src/gcc-build &&
./gcc-2.95.2/configure \
  --prefix=/usr/local/gcc2952 \
  --with-local-prefix=/usr/local/gcc2952 \
  --with-gxx-include-dir=/usr/local/gcc2952/include/g++ \
  --enable-shared --enable-languages=c,c++ &&
make bootstrap &&
make install
```

Command explanations

--with-local-prefix: GCC installs a number of files in /usr/local even when --prefix is set to something else. We don't want that to happen in this case so that's why we use the --with-local-prefix option to change that path.

--with-gxx-include-dir: GCC installs the C++ header files in /usr/include/g++ by default. Again, in this case we don't want that to happen, we want this GCC version to be installed completely under /usr/local/gcc2952.

make bootstrap: Compile GCC by bootstrapping it. Here that means the compiler will be built three times in total. First it is compiled with your system's default compiler (which will usually be a gcc or egcs compiler). This is stage 1 compiler. Then GCC will re-compile itself but instead of using your system's compiler it will use itself to compile itself again. This is the stage 2 compiler. Then it will compile itself a second time with the stage 2 compiler and compares the second and the third build to see if they are identical. If so, the compilation was a success.

Contents

The GCC package contains compilers, preprocessors and the GNU C++ Library.

Description

Compiler

A compiler translates source code in text format to a format that a computer understands. After a source code file is compiled into an object file, a linker will create an executable file from one or more of these compiler generated object files.

Pre-processor

A pre-processor pre-processes a source file, such as including the contents of header files into the source file. You generally don't do this yourself to save yourself a lot of time. You just insert a line like `#include <filename>`. The pre-processor file insert the contents of that file into the source file. That's one of the things a pre-processor does.

C++ Library

The C++ library is used by C++ programs. The C++ library contains functions that are frequently used in C++ programs. This way the programmer doesn't have to write certain functions (such as writing a string of text to the screen) from scratch every time he creates a program.

Installing GCC on the LFS system

Installation of GCC on the LFS system

After you unpacked the gcc-2.95.2 archive don't enter the newly created gcc-2.95.2 directory but stay in the \$LFS/usr/src directory. Install GCC by running the following commands:

```
mkdir $LFS/usr/src/gcc-build &&
cd $LFS/usr/src/gcc-build &&
../gcc-2.95.2/configure --prefix=/usr \
  --with-gxx-include-dir=/usr/include/g++ \
  --enable-languages=c,c++ --disable-nls &&
make -e LDFLAGS=-static bootstrap &&
make prefix=$LFS/usr local_prefix=$LFS/usr/local \
  gxx_include_dir=$LFS/usr/include/g++ install
```

Creating necessary symlinks

The system needs a few symlinks to ensure every program is able to find the compiler and the pre-processor. Some programs run the cc program, others run the gcc program. Some programs expect the cpp program in /lib and others expect to find it in /usr/bin. Create those symlinks by running:

```
cd $LFS/lib &&
ln -s ../usr/lib/gcc-lib/*/2.95.2/cpp cpp &&
cd $LFS/usr/lib &&
ln -s gcc-lib/*/2.95.2/cpp cpp &&
cd $LFS/usr/bin &&
ln -s gcc cc
```

Command explanations

--enable-languages=c,c++: This only builds the C and C++ compilers and not the other available compilers as they are, on the average, not often used. If you do need those other compilers don't use the --enable-languages parameter.

ln -s ../usr/lib/gcc-lib/*/2.95.2/cpp cpp: This creates the \$LFS/lib/cpp symlink. Some packages explicitly try to find cpp in /lib.

`ln -s ../usr/lib/gcc-lib/*/2.95.2/cpp cpp`: This creates the `$LFS/usr/lib/cpp` symlink as there are packages that expect `cpp` to be in `/usr/lib`.

Contents

The GCC package contains compilers, preprocessors and the GNU C++ Library.

Description

Compiler

A compiler translates source code in text format to a format that a computer understands. After a source code file is compiled into an object file, a linker will create an executable file from one or more of these compiler generated object files.

Pre-processor

A pre-processor pre-processes a source file, such as including the contents of header files into the source file. You generally don't do this yourself to save yourself a lot of time. You just insert a line like `#include <filename>`. The pre-processor file insert the contents of that file into the source file. That's one of the things a pre-processor does.

C++ Library

The C++ library is used by C++ programs. The C++ library contains functions that are frequently used in C++ programs. This way the programmer doesn't have to write certain functions (such as writing a string of text to the screen) from scratch every time he creates a program.

Installing Linux Kernel

Installation of Linux Kernel

We won't be compiling a new kernel image yet. We'll do that after we have finished the installation of the basic system software in this chapter. But because certain software need the kernel header files, we're going to unpack the kernel archive now and set it up so that we can compile package that need the kernel.

Create the kernel configuration file by running the following command:

```
yes "" | make config
```

Ignore the warning *Broken pipe* you might see at the end. Now run the following commands to set up all the dependencies correctly:

```
make dep
```

Now that that's done, we need to create the `$LFS/usr/include/linux` and the `$LFS/usr/include/asm` symlinks. Create them by running the following commands:

```
cd $LFS/usr/include &&  
ln -s ../src/linux/include/linux linux &&  
ln -s ../src/linux/include/asm asm
```

Command explanations

yes "" | make config: This runs `make config` and answers "Y" to every question the config script asks the user. We're not configuring the real kernel here, we just need to have some sort of configure file created so that we can run `make dep` next that will create a few files in `$LFS/usr/src/linux/include/linux` like `version.h` among others that we will need to compile Glibc and other packages later in chroot.

make dep: `make dep` checks dependencies and sets up the dependencies file. We don't really care about the dependency checks, but what we do care about is that `make dep` creates those aforementioned files in `$LFS/usr/src/linux/include/linux` we will be needing later on.

```
ln -s ../src/linux/include/linux linux and ln -s ../src/linux/include/asm  
asm:
```

These commands create the linux and asm symlinks in the \$LFS/usr/include directory that point to the proper directories in the Linux source tree. Packages that need kernel headers include them with lines like `#include <linux/errno.h>`. These paths are relative to the /usr/include directory so the /usr/include/linux link points to the directory containing the Linux kernel header files. The same goes for the asm symlink.

Contents

The Linux kernel package contains the Linux kernel.

Description

The Linux kernel is at the core of every Linux system. It's what makes Linux tick. When you turn on your computer and boot a Linux system, the very first piece of Linux software that gets loaded is the kernel. The kernel initializes the system's hardware components such as serial ports, parallel ports, sound cards, network cards, IDE controllers, SCSI controllers and a lot more. In a nutshell the kernel makes the hardware available so that the software can run.

Installing Glibc

Installation of Glibc

Unpack the `glibc-crypt` and `glibc-linuxthreads` in the `glibc-2.1.3` directory, not in `$LFS/usr/src`. Don't enter the created directories. Just unpack them and leave it with that.

A few default parameters of Glibc need to be changed, such as the directory where the shared libraries are supposed to be installed in and the directory that contains the system configuration files. For this purpose you need to create the `$LFS/usr/src/glibc-build` directory and `cd` into that directory with:

```
mkdir $LFS/usr/src/glibc-build &&  
cd $LFS/usr/src/glibc-build
```

In that directory you create a new file `configparms` by running the following:

```
cat > configparms << "EOF"  
# Begin configparms  
  
slibdir=/lib  
sysconfdir=/etc  
  
# End configparms  
EOF
```

Before we actually install Glibc you need to unpack the Glibc patch file.

Please note that the configure script of Glibc may complain about certain files in the `/usr/include` directory being too old and will be replaced, or that some symlink is not supposed to be there anymore (like the `/usr/include/scsi` symlink that's present on older Linux systems). If it asks you to move a symlink like `scsi` out of the way, please do so. If it says it will replace old files by the newer Glibc files you can ignore that. Glibc does not know that it will end up on `$LFS` when the configure script is run.

If your system had already a suitable GCC version installed, change to the `$LFS/usr/src/glibc-build` directory and install Glibc by running the following commands:

```
cd ../glibc-2.1.3 &&  
patch -Np1 -i ../glibc-2.1.3.patch &&  
cd $LFS/usr/src/glibc-build &&
```

```

../glibc-2.1.3/configure \
  --prefix=/usr --enable-add-ons \
  --with-headers=$LFS/usr/include  &&
make &&
make install_root=$LFS install &&
make install_root=$LFS localedata/install-locales

```

If your system didn't have a suitable GCC version installed, change to the `$LFS/usr/src/glibc-build` directory and install Glibc using the `gcc-2.95.2` you just installed by running the following commands:

```

cd ../glibc-2.1.3 &&
patch -Np1 -i ../glibc-2.1.3.patch &&
cd $LFS/usr/src/glibc-build &&
CC=/usr/local/gcc2952/bin/gcc \
  ../glibc-2.1.3/configure --prefix=/usr --enable-add-ons \
  --with-headers=$LFS/usr/include  &&
make &&
make install_root=$LFS install &&
make install_root=$LFS localedata/install-locales

```

Copying old NSS library files

If your normal Linux system runs `glibc-2.0`, you need to copy the NSS library files to the LFS partition. Certain statically linked programs still depend on the NSS library, especially programs that need to lookup usernames, userids and groupids. You can check which C library version your normal Linux system uses by running:

```
strings /lib/libc* | grep "release version"
```

The output of that command should tell you something like this:

```
GNU C Library stable release version 2.1.3, by Roland McGrath et al.
```

If you have `Glibc-2.0.x` installed on your starting distribution, copy the NSS library files by running:

```
cp -av /lib/libnss* $LFS/lib
```

Command explanations

patch -Np1 -i ../glibc-2.1.3.patch: This applies a patch that fixes a minor bug in Glibc. Glibc defines a few variables names with illegal characters in the name. Bash-2.03 and older don't complain about that but Bash-2.04 does and won't compile Glibc properly.

--enable-add-ons: This enabled the add-ons that we install with Glibc: linuxthreads and crypt.

--with-headers=\$LFS/usr/include: This makes Glibc use the kernel header files on our LFS system and not the kernel header files from your starting distribution which may be out-of-date or modified.

make install_root=\$LFS: This is the Glibc way to specify the equivalent of `--prefix=`.

Contents

The Glibc package contains the GNU C Library.

Description

The C Library is a collection of commonly used functions in programs. This way a programmer doesn't need to create his own functions for every single task. The most common things like writing a string to your screen are already present and at the disposal of the programmer.

The C library (actually almost every library) come in two flavours: dynamic ones and static ones. In short when a program uses a static C library, the code from the C library will be copied into the executable file. When a program uses a dynamic library, that executable will not contain the code from the C library, but instead a routine that loads the functions from the library at the time the program is run. This means a significant decrease in the file size of a program. If you don't understand this concept, you better read the documentation that comes with the C Library as it is too complicated to explain here in one or two lines.

Installing Grep

Installation of Grep

Install Grep by running the following commands:

```
CPPFLAGS=-Dre_max_failures=re_max_failures2 \  
./configure --prefix=$LFS/usr --disable-nls  &&  
make LDFLAGS=-static &&  
make install
```

Contents

The grep package contains the egrep, fgrep and grep programs.

Description

egrep

egrep prints lines from files matching an extended regular expression pattern.

fgrep

fgrep prints lines from files matching a list of fixed strings, separated by newlines, any of which is to be matched.

grep

grep prints lines from files matching a basic regular expression pattern.

Installing Gzip

Installation of Gzip

Before you install Gzip you have to unpack the gzip patch file.

```
patch -Np1 -i ../gzip-1.2.4a.patch &&
./configure --prefix=$LFS/usr --disable-nls &&
make LDFLAGS=-static &&
make install &&
cp $LFS/usr/bin/gunzip $LFS/usr/bin/gzip $LFS/bin &&
rm $LFS/usr/bin/gunzip $LFS/usr/bin/gzip
```

Contents

The Gzip package contains the gunzip, gzexe, gzip, zcat, zcmp, zdiff, zforce, zgrep, zmore and znew programs.

Description

gunzip

gunzip decompresses files that are compressed with gzip.

gzexe

gzexe allows you to compress executables in place and have them automatically uncompress and execute when you run them (at a penalty in performance).

gzip

gzip reduces the size of the named files using Lempel–Ziv coding (LZ77).

zcat

zcat uncompresses either a list of files on the command line or its standard input and writes the uncompressed data on standard output

zcmp

zcmp invokes the cmp program on compressed files.

zdiff

zdiff invokes the diff program on compressed files.

zforce

zforce forces a .gz extension on all gzip files so that gzip will not compress them twice. This can be useful for files with names truncated after a file transfer.

zgrep

zgrep invokes the grep program on compressed files.

zmore

Zmore is a filter which allows examination of compressed or plain text files one screenful at a time on a soft-copy terminal (similar to the more program).

znew

Znew recompresses files from .Z (compress) format to .gz (gzip) format.

Installing Make

Installation of Make

Install Make by running the following commands:

```
./configure --prefix=$LFS/usr --disable-nls &&  
make LDFLAGS=-static &&  
make install
```

Contents

The Make package contains the make program.

Description

make determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them.

Installing Sed

Installation of Sed

Install Sed by running the following commands:

```
CPPFLAGS=-Dre_max_failures=re_max_failures2 \  
./configure --prefix=$LFS/usr --disable-nls \  
--bindir=$LFS/bin && \  
make LDFLAGS=-static && \  
make install
```

Contents

The Sed package contains the sed program.

Description

sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline).

Installing Shellutils

Installation of Sh-utils

Install Shellutils by running the following commands:

```
./configure --prefix=$LFS/usr --disable-nls &&  
make LDFLAGS=-static &&  
make install &&  
cd $LFS/usr/bin &&  
mv date echo false pwd stty $LFS/bin &&  
mv su true uname hostname $LFS/bin
```

Contents

The Shellutils package contains the basename, chroot, date, dirname, echo, env, expr, factor, false, groups, hostid, hostname, id, logname, nice, nohup, pathchk, pinky, printenv, printf, pwd, seq, sleep, stty, su, tee, test, true, tty, uname, uptime, users, who, whoami and yes programs.

Description

basename

basename strips directory and suffixes from filenames.

chroot

chroot runs a command or interactive shell with special root directory.

date

date displays the current time in a specified format, or sets the system date.

dirname

dirname strips non-directory suffixes from file name.

echo

echo displays a line of text.

env

env runs a program in a modified environment.

expr

expr evaluates expressions.

factor

factor prints the prime factors of all specified integer numbers.

false

false always exits with a status code indicating failure.

groups

groups prints the groups a user is in.

hostid

hostid prints the numeric identifier (in hexadecimal) for the current host.

hostname

hostname sets or prints the name of the current host system

id

id prints the real and effective UIDs and GIDs of a user or the current user.

logname

logname prints the current user's login name.

nice

nice runs a program with modified scheduling priority.

nohup

nohup runs a command immune to hangups, with output to a non-tty

pathchk

pathchk checks whether file names are valid or portable.

pinky

pinky is a lightweight finger utility which retrieves information about a certain user

printenv

printenv prints all or part of the environment.

printf

printf formats and print data (the same as the printf C function).

pwd

pwd prints the name of the current/working directory

seq

seq prints numbers in a certain range with a certain increment.

sleep

sleep delays for a specified amount of time.

stty

stty changes and prints terminal line settings.

su

su runs a shell with substitute user and group IDs

tee

tee reads from standard input and write to standard output and files.

test

test checks file types and compares values.

true

True always exitx with a status code indicating success.

tty

tty prints the file name of the terminal connected to standard input.

uname

uname prints system information.

uptime

uptime tells how long the system has been running.

users

users prints the user names of users currently logged in to the current host.

who

who shows who is logged on.

whoami

whoami prints your effective userid.

yes

yes outputs a string repeatedly until killed.

Installing Tar

Installation of Tar

If you want to apply the Bzip2 tar patch which gives you the `-y` option to tar so you can use bzip2 files with tar, first download the patch from <http://sourceware.cygnum.com/bzip2/> and apply it by running the following command within the src directory under the tar-1.13 directory:

```
cd src &&  
patch -i ../../gnutarpatch.txt &&  
cd ..
```

Install Tar by running the following commands:

```
./configure --prefix=$LFS/usr --disable-nls &&  
make LDFLAGS=-static &&  
make prefix=$LFS/usr install &&  
mv $LFS/usr/bin/tar $LFS/bin
```

Contents

The tar package contains the tar and rmt programs.

Description

tar

tar is an archiving program designed to store and extract files from an archive file known as a tarfile.

rmt

rmt is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection.

Installing Textutils

Installation of Textutils

Install Textutils by running the following commands:

```
./configure --prefix=$LFS/usr --disable-nls &&  
make LDFLAGS=-static &&  
make install &&  
mv $LFS/usr/bin/cat $LFS/bin
```

Contents

The Textutils package contains the cat, cksum, comm, split, cut, expand, fmt, fold, head, join, md5sum, nl, od, paste, pr, ptx, sort, split, sum, tac, tail, tr, tsort, unexpand, uniq and wc programs.

Description

cat

cat concatenates file(s) or standard input to standard output.

cksum

cksum prints CRC checksum and byte counts of each specified file.

comm

comm compares two sorted files line by line.

csplit

csplit outputs pieces of a file separated by (a) pattern(s) to files xx01, xx02, ..., and outputs byte counts of each piece to standard output.

cut

cut prints selected parts of lines from specified files to standard output.

expand

expand converts tabs in files to spaces, writing to standard output.

fmt

fmt reformats each paragraph in the specified file(s), writing to standard output.

fold

fold wraps input lines in each specified file (standard input by default), writing to standard output.

head

Print first xx (10 by default) lines of each specified file to standard output.

join

join joins lines of two files on a common field.

md5sum

md5sum prints or checks MD5 checksums.

nl

nl writes each specified file to standard output, with line numbers added.

od

od writes an unambiguous representation, octal bytes by default, of a specified file to standard output.

paste

paste writes lines consisting of the sequentially corresponding lines from each specified file, separated by TABs, to standard output.

pr

pr paginates or columnates files for printing.

ptx

ptx produces a permuted index of file contents.

sort

sort writes sorted concatenation of files to standard output.

split

split outputs fixed-size pieces of an input file to PREFIXaa, PREFIXab, ...

sum

sum prints checksum and block counts for each specified file.

tac

tac writes each specified file to standard output, last line first.

tail

tail print the last xx (10 by default) lines of each specified file to standard output.

tr

tr translates, squeezes, and/or deletes characters from standard input, writing to standard output.

tsort

tsort writes totally ordered lists consistent with the partial ordering in specified files.

unexpand

unexpand converts spaces in each file to tabs, writing to standard output.

uniq

uniq discards all but one of successive identical lines from files or standard input and writes to files or standard output.

wc

wc prints line, word, and byte counts for each specified file, and a total line if more than one file is specified.

Creating passwd and group files

In order for user and group root to be recognized and to be able to logon it needs an entry in the `/etc/passwd` and `/etc/group` file. Besides the group root a couple of other groups are recommended and needed by packages. The groups with their GID's below aren't part of any standard. The LSB only recommends besides a group root a group bin to be present with GID 1. Other group names and GID's can be chosen by yourself. Well written packages don't depend on GID numbers but just use the group name, it doesn't matter all that much what GID a group has. Since there aren't any standards for groups I won't follow any conventions used by Debian, RedHat and others. The groups added here are the groups the MAKEDEV script (the script that creates the device files in the `/dev` directory) mentions.

Create a new file `$LFS/etc/passwd` by running the following command:

```
echo "root:x:0:0:root:/root:/bin/bash" > $LFS/etc/passwd
```

Create a new file `$LFS/etc/group` by running the following:

```
cat > $LFS/etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
uucp:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
EOF
```

Mounting \$LFS/proc file system

In order for certain programs to function properly the proc file system must be mounted and available from within the chroot'ed environment as well. It's not a problem to mount the proc file system twice or even more than that, since it's a virtual file system maintained by the kernel itself.

Mount the proc file system under \$LFS/proc by running the following command:

```
mount proc $LFS/proc -t proc
```

Chapter 6. Installing basic system software

Introduction

The installation of all the software is pretty straightforward and you'll think it's so much easier and shorter to give the generic installation instructions for each package and only explain how to install something if a certain package requires an alternate installation method. Although I agree with you on that, I, however, choose to give the full instructions for each and every package. This is simply to avoid any possible confusion and errors.

Debugging symbols and compiler optimizations

Most programs and libraries by default are compiled with debugging symbols and optimizing level 2 (gcc options `-g` and `-O2`) and are compiled for a specific CPU. On Intel platforms software is compiled for i386 processors by default. If you don't wish to run software on other machines other than your own, you might want to change the default compiler options so that they will be compiled with a higher optimization level, no debugging symbols and generate code for your specific architecture. Let me first explain what debugging symbols are.

A program compiled with debugging symbols means you can run a program or library through a debugger and the debugger's output will be user friendlier. These debugging symbols also enlarge the program or library significantly.

To remove debugging symbols from a binary (must be an a.out or ELF binary) run **strip** `--strip-debug filename` You can use wild cards if you need to strip debugging symbols from multiple files (use something like `strip --strip-debug $LFS/usr/bin/*`). Another, easier, options is just not to compile programs with debugging symbols. Most people will probably never use a debugger on software, so by leaving those symbols out you can save a lot of disk space.

Before you wonder if these debugging symbols would make a big difference, here are some statistics:

- A dynamic Bash binary with debugging symbols: 1.2MB
- A dynamic Bash binary without debugging symbols: 478KB
- `/lib` and `/usr/lib` (glibc and gcc files) with debugging symbols: 87MB
- `/lib` and `/usr/lib` (glibc and gcc files) without debugging symbols: 16MB

Sizes may vary depending on which compiler was used and which C library version was used to link dynamic programs against, but your results will be similar if you compare programs with and without debugging symbols. After I was done with this chapter and stripped all debugging symbols from all LFS binaries and libraries I regained a little over 102 MB of disk space. Quite the difference.

There are a few ways to change the default compiler options. One way is to edit every Makefile file you can find in a package, look for the `CFLAGS` and `CXXFLAGS` variables (a well designed package uses the `CFLAGS` variable to define gcc compiler options and `CXXFLAGS` to define g++ compiler options) and change their values. Packages like `binutils`, `gcc`, `glibc` and others have a lot of Makefile files in a lot of subdirectories so this would take a lot of time to do. Instead there's an easier way to do things: create the `CFLAGS` and `CXXFLAGS` environment variables. Most configure scripts read the `CFLAGS` and `CXXFLAGS` variables and use them in the Makefile files. A few packages don't follow this convention and those package require manual editing.

In the next section we'll create the `$LFS/root/.bash_profile` that will contain the following optimization:

```
CFLAGS="-O3 -march=xxx"  
CXXFLAGS=$CFLAGS
```

This is a minimal set of optimizations that ensures it works on almost all platforms. This option (march) will compile the binaries with specific instructions for that CPU you have specified. This means you can't copy this binary to a lower class CPU and execute it. It will either work very unreliable or not at all (it will give errors like Illegal Instruction, core dumped). You'll have to read the [GCC Info](#) page to find more possible optimization flags. In the above environment variable you have to replace xxx and yyy with the appropriate CPU identifiers such as i586, i686, powerpc and others.

Please keep in mind that if you find that a package doesn't compile and gives errors like "segmentation fault, core dumped" it's most likely got to do with these compiler optimizations. Try lowering the optimizing level by changing -O3 to -O2. If that doesn't work try -O or leave it out all together. Also try changing the -march variable. Compilers are very sensitive to certain hardware too. Bad memory can cause compilation problems when a high level of optimization is used, like the -O3 setting. The fact that I don't have any problems compiling everything with -O3 doesn't mean you won't have any problems either. Another problem can be the Binutils version that's installed on your system which often causes compilation problems in Glibc (most noticeable in RedHat because RedHat often uses beta software which aren't always very stable. "RedHat likes living on the bleeding edge, but leaves the bleeding up to you" (quoted from somebody on the lfs-discuss mailinglist).

Creating `$LFS/root/.bash_profile`

When we have entered the chroot'ed environment in the next section we want to export a couple of environment variables in that shell such as `TERM`, `CFLAGS`, `CXXFLAGS` and others variables you want to have set. For that purpose we'll create the `$LFS/root/.bash_profile` file and "source" it after we have entered the chroot'ed environment.

Create a new file `$LFS/root/.bash_profile` by running the following. Replace `-march=xxx` with the proper value for your machine:

```
cat > $LFS/root/.bash_profile << "EOF"
# Begin /root/.bash_profile

alias ls='ls -a --color'

PS1='\u:\w\$ '
TERM=linux
CFLAGS="-O3 -march=xxx"
CXXFLAGS=$CFLAGS

export PS1 TERM CFLAGS CXXFLAGS

# End /root/.bash_profile
EOF
```

You can add more environment variables at your own discretion as you deem them necessary.

Entering the chroot'ed environment

It's time to enter our chroot'ed environment in order to install the rest of the software we need.

Enter the following command to enter the chroot'ed environment. From this point on there's no need to use the `$LFS` variable anymore, because everything you do will be restricted to the LFS partition (since `/` is actually `/mnt/lfs` but the shell doesn't know that).

```
cd $LFS &&  
chroot $LFS env -i HOME=/root bash --login
```

Now that we are inside a chroot'ed environment, we can continue to install all the basic system software. Make sure you execute all the following commands in this chapter from within the chroot'ed environment.

Setting up locale

If you plan on using a non-English language you have to setup your locale to match your desired language. You do that by running the `localedef` command with the proper values. You can find the desired files in the `/usr/share/i18n` directory.

```
localedef -i en_US -f ISO-8859-1 en_US
```

Creating device files

Installation of MAKEDEV

Install MAKEDEV by running the following commands:

```
sed "s/# 9/9/" MAKEDEV >/dev/MAKEDEV &&  
chmod 755 /dev/MAKEDEV
```

Creating the /dev entries

Create the device files by running the following commands:

```
cd /dev &&  
./MAKEDEV -v generic
```

The "generic" parameter passed to the MAKEDEV script doesn't create all the devices you might need, such as audio devices, hdc, hdd and others. If you seem to be missing something tell MAKEDEV to create it. To create hdc replace generic with hdc. You can also add hdc to generic, so you would execute **./MAKEDEV -v generic hdc** to create the generic set of devices files, plus the files you need to be able to access hdc (and hdc1, hdc2, etc)

Please note that this script dates back from 1997 and therefore can be outdated and not support newer hardware. If you need device files which aren't known by this script please read the Documentation/devices.txt file in a Linux source tree. This file lists all the major and minor numbers for all the device files that the kernel knows about. With this list you can create such device files yourself. See the mknod man page for more information on how to make device files yourself.

Command explanations

sed "s/# 9/9/" MAKEDEV >/dev/MAKEDEV: By default the Makedev script only creates the hda1–hda8 and hdb1–hdb8 devices. By replacing "# 9" by "9"s in the MAKEDEV script, it will create hda1–hda20, hdb1–hdb20 and possible others (like hdc and hdd)

chmod 755 /dev/MAKEDEV: This sets the permissions of the MAKEDEV script to mode 754 which makes it executable only for owner and group and readable by everybody.

Contents

The MAKEDEV package contains the MAKEDEV script.

Description

MAKEDEV is a script that can aid you in creating the necessary static device files that usually reside in the /dev directory.

Installing Ed

Installation of Ed

Install Ed by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install &&  
mv /usr/bin/ed /usr/bin/red /bin
```

Contents

The Ed package contains the ed program.

Description

Ed is a line-oriented text editor. It is used to create, display, modify and otherwise manipulate text files.

Installing Patch

Installation of Patch

Install Patch by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install
```

Contents

The Patch package contains the patch program.

Description

The patch program modifies a file according to a patch file. A patch file usually is a list created by the diff program that contains instructions on how an original file needs to be modified. Patch is used a lot for source code patches since it saves time and space. Imagine you have a package that is 1MB in size. The next version of that package only has changes in two files of the first version. You can ship an entirely new package of 1MB or provide a patch file of 1KB which will update the first version to make it identical to the second version. So if you have downloaded the first version already, a patch file can save you a second large download.

Installing Findutils

Installing Findutils

Before you install Findutils you have to unpack the findutils patch file.

Install Findutils by running the following commands:

```
patch -Np1 -i ../findutils-4.1.patch &&  
./configure --prefix=/usr &&  
make &&  
make install
```

Contents

The Findutils package contains the find, locate, updatedb and xargs programs.

Description

Find

The find program searches for files in a directory hierarchy which match a certain criteria. If no criteria is given, it lists all files in the current directory and its subdirectories.

Locate

Locate scans a database which contain all files and directories on a filesystem. This program lists the files and directories in this database matching a certain criteria. If you're looking for a file this program will scan the database and tell you exactly where the files you requested are located. This only makes sense if your locate database is fairly up-to-date else it will provide you with out-of-date information.

Updatedb

The updatedb program updates the locate database. It scans the entire file system (including other file system that are currently mounted unless you specify it not to) and puts every directory and file it finds into the database that's used by the locate program which retrieves this information. It's a good practice to update this database once a day so that you are ensured of a database that is up-to-date.

Xargs

The `xargs` command applies a command to a list of files. If you need to perform the same command on multiple files, you can create a file that contains all these files (one per line) and use `xargs` to perform that command on the list.

Installing Mawk

Installation of Mawk

Install Mawk by running the following commands:

```
./configure &&  
make &&  
make BINDIR=/usr/bin \  
    MANDIR=/usr/share/man/man1 install &&  
cd /usr/bin &&  
ln -s mawk awk
```

Contents

The Mawk package contains the mawk program.

Description

mawk

Mawk is an interpreter for the AWK Programming Language. The AWK language is useful for manipulation of data files, text retrieval and processing, and for prototyping and experimenting with algorithms.

Installing Ncurses

Installation of Ncurses

Install Ncurses by running the following commands:

```
./configure --prefix=/usr --libdir=/lib \  
  --with-shared --disable-termcap &&  
make &&  
make install &&  
cd /lib &&  
ln -s libncurses.a libcurses.a
```

Command explanations

--with-shared: This enables the build of the shared ncurses library files.

--disable-termcap: Disabled the compilation of termcap fallback support.

ln -s libncurses.a libcurses.a: This creates the /lib/libcurses.a symlink that for some reason isn't created during the libncurses installation.

Contents

The Ncurses package contains the ncurses, panel, menu and form libraries. It also contains the tic, infocmp, clear, tput, toe and tset programs.

Description

The libraries

The libraries that make up the Ncurses library are used to display text (often in a fancy way) on your screen. An example where ncurses is used is in the kernel's "make menuconfig" process. The libraries contain routines to create panels, menu's, form and general text display routines.

Tic

Tic is the terminfo entry-description compiler. The program translates a terminfo file from source format into the binary format for use with the ncurses library routines. Terminfo files contain information about the

capabilities of your terminal.

Infocmp

The `infocmp` program can be used to compare a binary terminfo entry with other terminfo entries, rewrite a terminfo description to take advantage of the `use=` terminfo field, or print out a terminfo description from the binary file (`term`) in a variety of formats (the opposite of what `tic` does).

clear

The `clear` program clears your screen if this is possible. It looks in the environment for the terminal type and then in the terminfo database to figure out how to clear the screen.

tput

The `tput` program uses the terminfo database to make the values of terminal-dependent capabilities and information available to the shell, to initialize or reset the terminal, or return the long name of the requested terminal type.

toe

The `toe` program lists all available terminal types by primary name with descriptions.

tset

The `Tset` program initializes terminals so they can be used, but it's not widely used anymore. It's provided for 4.4BSD compatibility.

Installing Vim

Installation of Vim

You need to unpack both the vim-rt and vim-src packages to install Vim. Both packages will unpack their files into the vim-5.7 directory. This won't overwrite any files from the other package. So it doesn't matter in which order you do it. Install Vim by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install &&  
cd /usr/bin &&  
ln -s vim vi
```

If you are planning on installing the X Window system on your LFS system, you might want to re-compile Vim after you have installed X. Vim comes with a nice GUI version of the editor which requires X and a few other libraries to be installed. For more information read the Vim documentation.

Contents

The Vim package contains the ctags, etags, ex, gview, gvim, rgview, rgvim, rview, rvim, view, vim, vintutor and xxd programs.

Description

ctags

ctags generate tag files for source code.

etags

etags does the same as ctags but it can generate cross reference files which list information about the various source objects found in a set of language files.

ex

ex starts vim in Ex mode.

gview

gview is the GUI version of view.

gvim

gvim is the GUI version of vim.

rgview

rgview is teh GUI version of rview.

rgvim

rgvim is the GUI version of rvim.

rview

rview is a restricted version of view. No shell commands can be started and Vim can't be suspended.

rvim

rvim is the restricted version of vim. No shell commands can be started and Vim can't be suspended.

view

view starts vim in read-only mode.

vim

vim starts vim in the normal, default way.

vimtutor

vimtutor starts the Vim tutor.

xxd

xxd makes a hexdump or does the reverse.

Installing GCC

Installation of GCC

After you unpacked the gcc-2.95.2 archive don't enter the newly created gcc-2.95.2 directory but stay in the /usr/src directory. Install GCC by running the following commands:

```
mkdir /usr/src/gcc-build &&
cd /usr/src/gcc-build &&
../gcc-2.95.2/configure --prefix=/usr \
  --with-gxx-include-dir=/usr/include/g++ \
  --enable-shared --enable-languages=c,c++ &&
make bootstrap &&
make install
```

Contents

The GCC package contains compilers, preprocessors and the GNU C++ Library.

Description

Compiler

A compiler translates source code in text format to a format that a computer understands. After a source code file is compiled into an object file, a linker will create an executable file from one or more of these compiler generated object files.

Pre-processor

A pre-processor pre-processes a source file, such as including the contents of header files into the source file. You generally don't do this yourself to save yourself a lot of time. You just insert a line like #include <filename>. The pre-processor file insert the contents of that file into the source file. That's one of the things a pre-processor does.

C++ Library

The C++ library is used by C++ programs. The C++ library contains functions that are frequently used in C++ programs. This way the programmer doesn't have to write certain functions (such as writing a string of text to the screen) from scratch every time he creates a program.

Installing Bison

Installation of Bison

Install Bison by running the following commands:

```
./configure --prefix=/usr \  
    --datadir=/usr/share/bison  &&  
make &&  
make install
```

Some programs don't know about bison and try to find the yacc program (bison is a (better) alternative for yacc). So to please those few programs out there we'll create a yacc script that calls bison and have it emulate yacc's output file name conventions).

Create a new file `/usr/bin/yacc` by running the following:

```
cat > /usr/bin/yacc << "EOF"  
#!/bin/sh  
# Begin /usr/bin/yacc  
  
/usr/bin/bison -y $*  
  
# End /usr/bin/yacc  
EOF  
chmod 755 /usr/bin/yacc
```

Command explanations

`--datadir=/usr/share/bison`: This install the bison grammar files in `/usr/share/bison` rather than `/usr/share`.

Contents

The Bison package contains the bison program.

Description

Bison is a parser generator, a replacement for YACC. YACC stands for Yet Another Compiler Compiler. What is Bison then? It is a program that generates a program that analyses the structure of a textfile. Instead of writing the actual program you specify how things should be connected and with those rules a program is constructed that analyses the textfile.

There are a lot of examples where structure is needed and one of them is the calculator.

Given the string :

$$1 + 2 * 3$$

You can easily come to the result 7. Why ? Because of the structure. You know how to interpret the string. The computer doesn't know that and Bison is a tool to help it understand by presenting the string in the following way to the compiler:

$$\begin{array}{c}
 + \\
 /\backslash \\
 * \quad 1 \\
 /\backslash \\
 2 \quad 3
 \end{array}$$

You start at the bottom of a tree and you come across the numbers 2 and 3 which are joined by the multiplication symbol, so the computer multiplies 2 and 3. The result of that multiplication is remembered and the next thing that the computer sees is the result of 2*3 and the number 1 which are joined by the add symbol. Adding 1 to the previous result makes 7. In calculating the most complex calculations can be broken down in this tree format and the computer just starts at the bottom and works its way up to the top and comes with the correct answer. Of course, Bison isn't only used for calculators alone.

Installing Less

Installation of Less

Install Less by running the following commands:

```
./configure --prefix=/usr --bindir=/bin &&  
make &&  
make install
```

Contents

The Less package contains the less program

Description

The less program is a file pager (or text viewer). It displays the contents of a file with the ability to scroll. Less is an improvement on the common pager called "more". Less has the ability to scroll backwards through files as well and it doesn't need to read the entire file when it starts, which makes it faster when you are reading large files.

Installing Groff

Installation of Groff

Install Groff by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install
```

Contents

The Groff packages contains the addftinfo, afmtodit, eqn, grodvi, groff, grog, grohtml, grolj4, grops, grotty, hpftodit, indxbib, lkbib, lookbib, neqn, nroff, pfbtops, pic, psbb, refer, soelim, tbl, tfmtodit and troff programs.

Description

addftinfo

addftinfo reads a troff font file and adds some additional font-metric information that is used by the groff system.

afmtodit

afmtodit creates a font file for use with groff and grops.

eqn

eqn compiles descriptions of equations embedded within troff input files into commands that are understood by troff.

grodvi

grodvi is a driver for groff that produces TeX dvi format.

groff

groff is a front-end to the groff document formatting system. Normally it runs the troff program and a postprocessor appropriate for the selected device.

grog

grog reads files and guesses which of the groff options `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s`, and `-t` are required for printing files, and prints the groff command including those options on the standard output.

grohtml

grohtml translates the output of GNU troff to html

grolj4

grolj4 is a driver for groff that produces output in PCL5 format suitable for an HP Laserjet 4 printer.

grops

grops translates the output of GNU troff to PostScript.

grotty

grotty translates the output of GNU troff into a form suitable for typewriter-like devices.

hpftodit

hpftodit creates a font file for use with groff `-Tlj4` from an HP tagged font metric file.

indxbib

indxbib makes an inverted index for the bibliographic databases a specified file for use with refer, lookbib, and lkbib.

lkbib

lkbib searches bibliographic databases for references that contain specified keys and prints any references found on the standard output.

lookbib

lookbib prints a prompt on the standard error (unless the standard input is not a terminal), reads from the standard input a line containing a set of keywords, searches the bibliographic databases in a specified file for references containing those keywords, prints any references found on the standard output, and repeats this process until the end of input.

neqn

It is currently not known what neqn is and what it does.

nroff

The nroff script emulates the nroff command using groff.

pfbtops

pfbtops translates a PostScript font in .pfb format to ASCII.

pic

pic compiles descriptions of pictures embedded within troff or TeX input files into commands that are understood by TeX or troff.

psbb

psbb reads a file which should be a PostScript document conforming to the Document Structuring conventions and looks for a %%BoundingBox comment.

refer

refer copies the contents of a file to the standard output, except that lines between .[and .] are interpreted as citations, and lines between .R1 and .R2 are interpreted as commands about how citations are to be processed.

soelim

soelim reads files and replaces lines of the form *.so file* by the contents of *file*.

tbl

tbl compiles descriptions of tables embedded within troff input files into commands that are understood by troff.

tfmtoedit

tfmtoedit creates a font file for use with **groff -Tdvi**

troff

troff is highly compatible with Unix troff. Usually it should be invoked using the groff command, which will also run preprocessors and postprocessors in the appropriate order and with the appropriate options.

Installing Man

Installation of Man

Install Man by running the following commands:

```
./configure -default &&  
make &&  
make install &&  
sed s/AWK="/AWK=\usr\bin\mawk"/ /usr/sbin/makewhatis >  
makewhatis-new &&  
mv makewhatis-new /usr/sbin/makewhatis &&  
chmod 755 /usr/sbin/makewhatis
```

Command explanations

-default: This configures the man package with default settings.

sed s/AWK="/AWK=\usr\bin\mawk"/ /usr/sbin/makewhatis >
makewhatis-new: This modifies /usr/sbin/makewhatis's AWK variable and fills in the location of the mawk program.

chmod 755 /usr/sbin/makewhatis: This makes the makewhatis script executable again.

Contents

The Man package contains the man, apropos whatis and makewhatis programs.

Description

man

man formats and displays the on-line manual pages.

apropos

apropos searches a set of database files containing short descriptions of system commands for keywords and displays the result on the standard output.

whatis

`whatis` searches a set of database files containing short descriptions of system commands for keywords and displays the result on the standard output. Only complete word matches are displayed.

makewhatis

`makewhatis` reads all the manual pages contained in given sections of `manpath` or the preformatted pages contained in the given sections of `catpath`. For each page, it writes a line in the `whatis` database; each line consists of the name of the page and a short description, separated by a dash. The description is extracted using the content of the `NAME` section of the manual page.

Installing Perl

Installation of Perl

Install Perl by running the following commands:

```
./Configure -Dprefix=/usr &&  
make &&  
make install
```

If you don't want to answer all those questions Perl asks you, you can add the `-d` option to the configure script and Perl will use all the default settings. To avoid the Configure script asking you questions after the `config.sh` file has been created you can pass the `-e` parameter to perl as well. The commands with these parameters included will be:

```
./Configure -Dprefix=/usr -d -e &&  
make &&  
make install
```

Contents

The Perl package contains Perl – Practical Extraction and Report Language

Description

Perl combines the features and capabilities of C, awk, sed and sh into one powerful programming language.

Installing M4

Installation of M4

Install M4 by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install
```

If your base system is running a 2.0 kernel and your Glibc version is 2.1 then you will most likely get problems executing M4 in the chroot'ed environment due to incompatibilities between the M4 program, Glibc-2.1 and the running 2.0 kernel. If you have problems executing the m4 program in the chroot'ed environment (for example when you install the autoconf and automake packages) you'll have to exit the chroot'ed environment and compile M4 statically. This way the binary is linked against Glibc 2.0 (if you run kernel 2.0 you're Glibc version is 2.0 as well on a decent system. Kernel 2.0 and Glibc-2.1 don't mix very well) and won't give you any problems.

To create a statically linked version of M4, execute the following commands:

```
logout  
cd $LFS/usr/src/m4-1.4  
./configure --prefix=/usr --disable-nls  
make LDFLAGS=-static  
make prefix=$LFS/usr install
```

Now you can re-enter the chroot'ed environment and continue with the next package. If you wish to recompile M4 dynamically, you can do that after you have rebooted into the LFS system rather than chroot'ed into it.

```
chroot $LFS env -i HOME=/root bash --login
```

Contents

The M4 package contains the M4 processor

Description

M4 is a macro processor. It copies input to output expanding macros as it goes. Macros are either builtin or user-defined and can take any number of arguments. Besides just doing macro expansion m4 has builtin functions for including named files, running UNIX commands, doing integer arithmetic, manipulating text in various ways, recursion, etc. M4 can be used either as a front-end to a compiler or as a macro processor in its own right.

Installing Texinfo

Installation of Texinfo

Install Texinfo by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install
```

Contents

The Texinfo package contains the info, install-info, makeinfo, texi2dvi and texindex programs

Description

info

The info program reads Info documents, usually contained in your /usr/doc/info directory. Info documents are like man(ual) pages, but they tend to be more in depth than just explaining the options to a program.

install-info

The install-info program updates the info entries. When you run the info program a list with available topics (ie: available info documents) will be presented. The install-info program is used to maintain this list of available topics. If you decide to remove info files manually, you need to delete the topic in the index file as well. This program is used for that. It also works the other way around when you add info documents.

makeinfo

The makeinfo program translates Texinfo source documents into various formats. Available formats are: info files, plain text and HTML.

texi2dvi

The texi2dvi program prints Texinfo documents

texindex

The texindex program is used to sort Texinfo index files.

Installing Autoconf

Installation of Autoconf

Install Autoconf by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install
```

Contents

The Autoconf package contains the autoconf, autoheader, autoreconf, autoscan, autoupdate and ifnames programs

Description

autoconf

Autoconf is a tool for producing shell scripts that automatically configure software source code packages to adapt to many kinds of UNIX-like systems. The configuration scripts produced by Autoconf are independent of Autoconf when they are run, so their users do not need to have Autoconf.

autoheader

The autoheader program can create a template file of C #define statements for configure to use

autoreconf

If you have a lot of Autoconf-generated configure scripts, the autoreconf program can save you some work. It runs autoconf (and autoheader, where appropriate) repeatedly to remake the Autoconf configure scripts and configuration header templates in the directory tree rooted at the current directory.

autoscan

The autoscan program can help you create a configure.in file for a software package. autoscan examines source files in the directory tree rooted at a directory given as a command line argument, or the current

directory if none is given. It searches the source files for common portability problems and creates a file `configure.scan` which is a preliminary `configure.in` for that package.

autoupdate

The `autoupdate` program updates a `configure.in` file that calls `Autoconf` macros by their old names to use the current macro names.

ifnames

`ifnames` can help when writing a `configure.in` for a software package. It prints the identifiers that the package already uses in C preprocessor conditionals. If a package has already been set up to have some portability, this program can help you figure out what its `configure` needs to check for. It may help fill in some gaps in a `configure.in` generated by `autoscan`.

Installing Automake

Installation of Automake

Install Automake by running the following commands:

```
./configure --prefix=/usr &&  
make install
```

Contents

The Automake package contains the `aclocal` and `automake` programs

Description

`aclocal`

Automake includes a number of Autoconf macros which can be used in your package; some of them are actually required by Automake in certain situations. These macros must be defined in your `aclocal.m4`; otherwise they will not be seen by autoconf.

The `aclocal` program will automatically generate `aclocal.m4` files based on the contents of `configure.in`. This provides a convenient way to get Automake–provided macros, without having to search around. Also, the `aclocal` mechanism is extensible for use by other packages.

`automake`

To create all the `Makefile.in`'s for a package, run the `automake` program in the top level directory, with no arguments. `automake` will automatically find each appropriate `Makefile.am` (by scanning `configure.in`) and generate the corresponding `Makefile.in`.

Installing Bash

Installation of Bash

Install Bash by running the following commands:

```
./configure --prefix=/usr --with-curses &&  
make &&  
make install &&  
logout
```

Replace the static bash with the dynamic bash and re-enter the chroot'ed environment by running:

```
mv $LFS/usr/bin/bash $LFS/usr/bin/bashbug $LFS/bin &&  
chroot $LFS env -i HOME=/root bash --login
```

Contents

The Bash package contains the bash program

Description

Bash is the Bourne–Again SHell, which is a widely used command interpreter on Unix systems. Bash is a program that reads from standard input, the keyboard. You type something and the program will evaluate what you have typed and do something with it, like running a program.

Installing Flex

Installation of Flex

Install Flex by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install
```

Contents

The Flex package contains the flex program

Description

Flex is a tool for generating programs which recognize patterns in text. Pattern recognition is very useful in many applications. You set up rules what to look for and flex will make a program that looks for those patterns. The reason people use flex is that it is much easier to set up rules for what to look for than to write the actual program that finds the text.

Installing File

Installation of File

Install File by running the following commands:

```
./configure --prefix=/usr --datadir=/usr/share/misc &&  
make &&  
make install
```

Contents

The File package contains the file program.

Description

File tests each specified file in an attempt to classify it. There are three sets of tests, performed in this order: filesystem tests, magic number tests, and language tests. The first test that succeeds causes the file type to be printed.

Installing Libtool

Installation of Libtool

Install Libtool by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install
```

Contents

The Libtool package contains the libtool and libtoolize programs. It also contains the ltdl library.

Description

libtool

Libtool provides generalized library-building support services.

libtoolize

libtoolize provides a standard way to add libtool support to your package.

ltdl library

Libtool provides a small library, called 'libltdl', that aims at hiding the various difficulties of dlopening libraries from programmers.

Installing Bin86

Installation of Bin86

Install Linux86 by running the following commands:

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

Contents

The Bin86 contains the as86, as86_encap, ld86, objdump86, nm86 and size86 programs.

Description

as86

as86 is an assembler for the 8086...80386 processors.

as86_encap

as86_encap is a shell script to call as86 and convert the created binary into a C file prog.v to be included in or linked with programs like boot block installers.

ld86

ld86 understands only the object files produced by the as86 assembler, it can link them into either an impure or a separate I&D executable.

objdump86

No description available.

nm86

No description available.

size86

No description available.

Installing Binutils

Installation of Binutils

Install Binutils by running the following commands:

```
./configure --prefix=/usr --enable-shared &&  
make -e tooldir=/usr &&  
make -e tooldir=/usr install
```

Description

The Binutils package contains the ld, as, ar, nm, objcopy, objdump, ranlib, size, strings, strip, c++filt, addr2line and nlmconv programs

Description

ld

ld combines a number of object and archive files, relocates their data and ties up symbol references. Often the last step in building a new compiled program to run is a call to ld.

as

as is primarily intended to assemble the output of the GNU C compiler gcc for use by the linker ld.

ar

The ar program creates, modifies, and extracts from archives. An archive is a single file holding a collection of other files in a structure that makes it possible to retrieve the original individual files (called members of the archive).

nm

nm lists the symbols from object files.

objcopy

objcopy utility copies the contents of an object file to another. objcopy uses the GNU BFD Library to read and write the object files. It can write the destination object file in a format different from that of the source object file.

objdump

objdump displays information about one or more object files. The options control what particular information to display. This information is mostly useful to programmers who are working on the compilation tools, as opposed to programmers who just want their program to compile and work.

ranlib

ranlib generates an index to the contents of an archive, and stores it in the archive. The index lists each symbol defined by a member of an archive that is a relocatable object file.

size

size lists the section sizes —and the total size— for each of the object files objfile in its argument list. By default, one line of output is generated for each object file or each module in an archive.

strings

For each file given, strings prints the printable character sequences that are at least 4 characters long (or the number specified with an option to the program) and are followed by an unprintable character. By default, it only prints the strings from the initialized and loaded sections of object files; for other types of files, it prints the strings from the whole file.

strings is mainly useful for determining the contents of non-text files.

strip

strip discards all or specific symbols from object files. The list of object files may include archives. At least one object file must be given. strip modifies the files named in its argument, rather than writing modified copies under different names.

c++filt

The C++ language provides function overloading, which means that you can write many functions with the same name (providing each takes parameters of different types). All C++ function names are encoded into a low-level assembly label (this process is known as mangling). The c++filt program does the inverse

mapping: it decodes (demangles) low-level names into user-level names so that the linker can keep these overloaded functions from clashing.

addr2line

addr2line translates program addresses into file names and line numbers. Given an address and an executable, it uses the debugging information in the executable to figure out which file name and line number are associated with a given address.

nlmconv

nlmconv converts relocatable object files into the NetWare Loadable Module files, optionally reading header files for NLM header information.

Installing Bzip2

Installation of Bzip2

Install Bzip2 by running the following commands:

```
make -f Makefile-libbz2_so &&
make bzip2recover libbz2.a &&
cp bzip2-shared /bin/bzip2 &&
cp bzip2recover /bin &&
cp bzip2.1 /usr/share/man/man1 &&
cp bzlib.h /usr/include &&
cp -a libbz2.so* libbz2.a /lib &&
rm /usr/lib/libbz2.a &&
cd /bin &&
rm bunzip2 && ln -s bzip2 bunzip2 &&
rm bzcata && ln -s bzip2 bzcata &&
cd /usr/share/man/man1 &&
ln -s bzip2.1 bunzip2.1 &&
ln -s bzip2.1 bzcata.1 &&
ln -s bzip2.1 bzip2recover.1
```

Although it's not strictly a part of a basic LFS system it's worth mentioning that you can download a patch for Tar which enables the tar program to compress and uncompress using bzip2/bunzip2 easily. With a plain tar you'll have to use constructions like `bzcat file.tar.bz|tar xv` or `tar --use-compress-prog=bunzip2 -xvf file.tar.bz2` to use bzip2 and bunzip2 with tar. This patch gives you the `-y` option so you can unpack a Bzip2 archive with `tar xvfy file.tar.bz2`. Applying this patch will be mentioned later on when you re-install the Tar package.

Command explanations

make -f Makefile-libbz2_so: This will cause bzip2 to be build using a different Makefile file, in this case the `Makefile-libbz2_so` file which creates a dynamic `libbz2.so` library and links the bzip2 utilities against it.

Contents

The Bzip2 packages contains the `bzip2`, `bunzip2`, `bzcata` and `bzip2recover` programs.

Description

Bzip2

bzip2 compresses files using the Burrows–Wheeler block sorting text compression algorithm, and Huffman coding. Compression is generally considerably better than that achieved by more conventional LZ77/LZ78–based compressors, and approaches the performance of the PPM family of statistical compressors.

Bunzip2

Bunzip2 decompresses files that are compressed with bzip2.

bzcat

bzcat (or bzip2 -dc) decompresses all specified files to the standard output.

bzip2recover

bzip2recover recovers data from damaged bzip2 files.

Installing Gettext

Installation of Gettext

Install Gettext by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install &&  
mv /po-mode.el /usr/share/gettext
```

Contents

The gettext package contains the gettext, gettextize, msgcmp, msgcomm, msgfmt, msgmerge, msgunfmt and xgettext programs.

Description

gettext

The gettext package is used for internationalization (also known as i18n) and for localization (also known as l10n). Programs can be compiled with Native Language Support (NLS) which enable them to output messages in your native language rather than in the default English language.

Installing Consoletools

Installation of Console-tools

Before you start installing Console-tools you have to unpack the console-tools-0.2.3.patch file.

Install Console-tools by running the following commands:

```
patch -Np1 -i ../console-tools-0.2.3.patch &&  
./configure --prefix=/usr &&  
make &&  
make install
```

Contents

The Console-tools package contains the charset, chvt, consolechars, dealloctv, dumpkeys, fgconsole, fix_bs_and_del, font2psf, getkeycodes, kbd_mode, loadkeys, loadunimap, mapscrn, mk_modmap, openvt, psfaddtable, psfgettable, psfstrietable, resizecons, saveunimap, screendump, setfont, setkeycodes, setleds, setmetamode, setvesablank, showfont, showkey, splitfont, unicode_start, unicode_stop, vctime, vt-is-URF8, writetv

Description

charset

charset sets an ACM for use in one of the G0/G1 charsets slots.

chvt

chvt changes foreground virtual terminal.

codepage

No description available.

consolechars

consolechars loads EGA/VGA console screen fonts, screen font maps and/or application–charset maps.

deallocvt

deallocvt deallocates unused virtual terminals.

dumpkeys

dumpkeys dumps keyboard translation tables.

fgconsole

fgconsole prints the number of the active virtual terminal.

fix_bs_and_del

No description available.

font2psf

No description available.

getkeycodes

getkeycodes prints the kernel scancode–to–keycode mapping table.

kbd_mode

kbd_mode reports or sets the keyboard mode.

loadkeys

loadkeys loads keyboard translation tables.

loadunimap

No description available.

mapscrn

No description available.

mk_modmap

No description available.

openvt

openvt starts a program on a new virtual terminal.

psfaddtable

psfaddtable adds a Unicode character table to a console font.

psfgettable

psfgettable extracts the embedded Unicode character table from a console font.

psfstriptime

psfstriptime removes the embedded Unicode character table from a console font.

resizecons

resizecons changes the kernel idea of the console size.

saveunimap

No description available.

screendump

No description available.

setfont

No description available.

setkeycodes

setkeycodes loads kernel scancode-to-keycode mapping table entries.

setleds

setleds sets the keyboard leds.

setmetamode

setmetamode defines the keyboard meta key handling.

setvesablank

No description available.

showfont

showfont displays all character in the current screenfont.

showkey

showkey examines the scancodes and keycodes sent by the keyboard.

splitfont

No description available.

unicode_start

unicode_start puts the console in Unicode mode.

unicode_stop

No description available.

vcstime

No description available.

vt-is-UTF8

vt-is-UTF8 checks whether the current virtual terminal is in UTF8- or byte-mode.

writetv

No description available.

Installing Consoledata

Installation of Console-data

Replace <path-to-kmap-file> below with the correct path to the desired kmap.gz file. An example could be i386/qwerty/us.kmap.gz

Install Console-data by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install &&  
cd /usr/share/keymaps &&  
ln -s <path-to-kmap-file> defkeymap.kmap.gz
```

Contents

The console-data package contains the data files that are used and needed by the console-tools package.

Installing Diffutils

Installation of Diffutils

Install Diffutils by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install
```

Contents

The Diffutils package contains the `cmp`, `diff`, `diff3` and `sdiff` programs.

Description

`cmp` and `diff`

`cmp` and `diff` both compare two files and report their differences. Both programs have extra options which compare files in different situations.

`diff3`

The difference between `diff` and `diff3` is that `diff` compares 2 files, `diff3` compares 3 files.

`sdiff`

`sdiff` merges two two files and interactively outputs the results.

Installing E2fsprogs

Installation of E2fsprogs

Install E2fsprogs by running the following commands:

Please note that the empty `--with-root-prefix=` option below is supposed to be like this. I did not forget to supply a value there.

```
./configure --prefix=/usr --with-root-prefix= \  
  --enable-elf-shlibs &&  
make &&  
make install &&  
make install-libs
```

Contents

The e2fsprogs package contains the `chattr`, `lsattr`, `uuidgen`, `badblocks`, `debugfs`, `dumpe2fs`, `e2fsck`, `e2label`, `fsck`, `fsck.ext2`, `mke2fs`, `mkfs.ext2`, `mklost+found` and `tune2fs` programs.

Description

chattr

`chattr` changes the file attributes on a Linux second extended file system.

lsattr

`lsattr` lists the file attributes on a second extended file system.

uuidgen

The `uuidgen` program creates a new universally unique identifier (UUID) using the `libuuid` library. The new UUID can reasonably be considered unique among all UUIDs created on the local system, and among UUIDs created on other systems in the past and in the future.

badblocks

badblocks is used to search for bad blocks on a device (usually a disk partition).

debugfs

The debugfs program is a file system debugger. It can be used to examine and change the state of an ext2 file system.

dumpe2fs

dumpe2fs prints the super block and blocks group information for the filesystem present on a specified device.

e2fsck and fsck.ext2

e2fsck is used to check a Linux second extended file system. fsck.ext2 does the same as e2fsck.

e2label

e2label will display or change the filesystem label on the ext2 filesystem located on the specified device.

fsck

fsck is used to check and optionally repair a Linux file system.

mke2fs and mkfs.ext2

mke2fs is used to create a Linux second extended file system on a device (usually a disk partition). mkfs.ext2 does the same as mke2fs.

mklost+found

mklost+found is used to create a lost+found directory in the current working directory on a Linux second extended file system. mklost+found pre-allocates disk blocks to the directory to make it usable by e2fsck.

tune2fs

tune2fs adjusts tunable filesystem parameters on a Linux second extended filesystem.

Installing Fileutils

Installation of Fileutils

Install Fileutils by running the following commands:

```
./configure --prefix=/usr --bindir=/bin &&  
make &&  
make install
```

Contents

The Fileutils package contains the chgrp, chmod, chown, cp, dd, df, dir, dircolors, du, install, ln, ls, mkdir, mkfifo, mknod, mv, rm, rmdir, sync, touch and vdir programs.

Description

chgrp

chgrp changes the group ownership of each given file to the named group, which can be either a group name or a numeric group ID.

chmod

chmod changes the permissions of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new permissions.

chown

chown changes the user and/or group ownership of each given file.

cp

cp copies files from one place to another.

dd

`dd` copies a file (from the standard input to the standard output, by default) with a user-selectable blocksize, while optionally performing conversions on it.

df

`df` displays the amount of disk space available on the filesystem containing each file name argument. If no file name is given, the space available on all currently mounted filesystems is shown.

ls, dir and vdir

`dir` and `vdir` are versions of `ls` with different default output formats. These programs list each given file or directory name. Directory contents are sorted alphabetically. For `ls`, files are by default listed in columns, sorted vertically, if the standard output is a terminal; otherwise they are listed one per line. For `dir`, files are by default listed in columns, sorted vertically. For `vdir`, files are by default listed in long format.

dircolors

`dircolors` outputs commands to set the `LS_COLOR` environment variable. The `LS_COLOR` variable is used to change the default color scheme used by `ls` and related utilities.

du

`du` displays the amount of disk space used by each argument and for each subdirectory of directory arguments.

install

`install` copies files and sets their permission modes and, if possible, their owner and group.

ln

`ln` makes hard or soft (symbolic) links between files.

mkdir

`mkdir` creates directories with a given name.

mkfifo

mkfifo creates a FIFO with each given name.

mknod

mknod creates a FIFO, character special file, or block special file with the given file name.

mv

mv moves files from one directory to another or renames files, depending on the arguments given to mv.

rm

rm removes files or directories.

rmdir

rmdir removes directories, if they are empty.

sync

sync forces changed blocks to disk and updates the super block.

touch

touch changes the access and modification times of each given file to the current time. Files that do not exist are created empty.

Installing Grep

Installation of Grep

Install Grep by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install
```

Contents

The grep package contains the egrep, fgrep and grep programs.

Description

egrep

egrep prints lines from files matching an extended regular expression pattern.

fgrep

fgrep prints lines from files matching a list of fixed strings, separated by newlines, any of which is to be matched.

grep

grep prints lines from files matching a basic regular expression pattern.

Installing Gzip

Installation of Gzip

Install Gzip by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install &&  
cd /usr/bin &&  
mv gzip /bin &&  
rm gunzip /bin/gunzip &&  
cd /bin &&  
ln -s gzip gunzip
```

Contents

The Gzip package contains the gunzip, gzexe, gzip, zcat, zcmp, zdiff, zforce, zgrep, zmore and znew programs.

Description

gunzip

gunzip decompresses files that are compressed with gzip.

gzexe

gzexe allows you to compress executables in place and have them automatically uncompress and execute when you run them (at a penalty in performance).

gzip

gzip reduces the size of the named files using Lempel–Ziv coding (LZ77).

zcat

zcat uncompresses either a list of files on the command line or its standard input and writes the uncompressed data on standard output

zcmp

zcmp invokes the cmp program on compressed files.

zdiff

zdiff invokes the diff program on compressed files.

zforce

zforce forces a .gz extension on all gzip files so that gzip will not compress them twice. This can be useful for files with names truncated after a file transfer.

zgrep

zgrep invokes the grep program on compressed files.

zmore

Zmore is a filter which allows examination of compressed or plain text files one screenful at a time on a soft-copy terminal (similar to the more program).

znew

Znew recompresses files from .Z (compress) format to .gz (gzip) format.

Installing Ldso

Installation of Ld.so

Install Ld.so by running the following commands:

```
cd util &&
make ldd ldconfig &&
cp ldd /bin &&
cp ldconfig /sbin &&
cd ../man &&
cp ldd.1 /usr/share/man/man1 &&
cp *.8 /usr/share/man/man8 &&
rm /usr/bin/ldd &&
hash -r
```

The "hash -r" command is to make bash forget about the locations of previously executed commands. If you have executed ldd before, bash expects it to be found in /usr/bin. Since we moved it to /bin, the cache needs to be purged so bash can find it in /bin when you want to execute it again.

You might have noticed that we don't use the compiler optimizations for this package. The reason is that overriding the CFLAGS variable causes compilation problems. You would have to edit the Config.mk file and add the proper values to the CFLAGS variable and then compile the package. If you want to do that it's up to you. I don't think it's worth the trouble though. The ld and ldd programs usually are only rarely used.

Contents

From the Ld.so package we're using the ldconfig and ldd programs.

Description

ldconfig

ldconfig creates the necessary links and cache (for use by the run-time linker, ld.so) to the most recent shared libraries found in the directories specified on the command line, in the file /etc/ld.so.conf, and in the trusted directories (/usr/lib and /lib). ldconfig checks the header and file names of the libraries it encounters when determining which versions should have their links updated.

ldd

ldd prints the shared libraries required by each program or shared library specified on the command line.

Installing Lilo

Installation of Lilo

Edit the `Makefile` file and edit the `CFLAGS` variable if you want to add compiler optimization to this package.

Install Lilo by running the following commands:

```
make &&  
make install
```

It appears that compilation of this package fails on certain machines when the `-g` compiler flag is being used. If you can't compile Lilo at all, please try removing the `-g` value from the `CFLAGS` variable in the `Makefile` file.

At the end of the installation the `make install` process will print a message stating that you have to execute `/sbin/lilo` to complete the update. Don't do this as it has no use. The `/etc/lilo.conf` isn't present yet. We will complete the installation of lilo in chapter 8.

Contents

The Lilo package contains the lilo program.

Description

lilo installs the Linux boot loader which is used to start a Linux system.

Installing Make

Installation of Make

Install Make by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install
```

Contents

The Make package contains the make program.

Description

make determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them.

Installing Modutils

Installation of Modutils

Install Modutils by running the following commands:

```
./configure &&  
make &&  
make install
```

Contents

The Modutils package contains the depmod, genksyms, insmod, insmod_ksymoops_clean, kerneld, kernelversion, ksyms, lsmod, modinfo, modprobe and rmmod programs.

Description

depmod

depmod handles dependency descriptions for loadable kernel modules.

genksyms

genksyms reads (on standard input) the output from gcc -E source.c and generates a file containing version information.

insmod

insmod installs a loadable module in the running kernel.

insmod_ksymoops_clean

insmod_ksymoops_clean deletes saved ksyms and modules not accessed in 2 days.

kerneld

kerneld performs kernel action in user space (such as on-demand loading of modules)

kernelversion

kernelversion reports the major version of the running kernel.

ksyms

ksyms displays exported kernel symbols.

lsmod

lsmod shows information about all loaded modules.

modinfo

modinfo examines an object file associated with a kernel module and displays any information that it can glean.

modprobe

Modprobe uses a Makefile-like dependency file, created by depmod, to automatically load the relevant module(s) from the set of modules available in predefined directory trees.

rmmod

rmmod unloads loadable modules from the running kernel.

Installing Procinfo

Installation of Procinfo

Edit the `Makefile` file and edit the `CFLAGS` variable if you want to add compiler optimizations to this package.

Install Procinfo by running the following commands:

```
sed "s/-ltermcap/-lncurses/" Makefile >Makefile2 &&  
make -f Makefile2 &&  
make install
```

Contents

The Procinfo package contains the `procinfo` program.

Description

`procinfo` gathers some system data from the `/proc` directory and prints it nicely formatted on the standard output device.

Installing Procps

Installation of Procps

Edit the `Makefile` file and edit the `CFLAGS` variable if you want to add compiler optimizations to this package.

Install Procps by running the following commands:

```
sed "s/XConsole/#XConsole/" Makefile >Makefile2  &&  
make -f Makefile2 &&  
make -f Makefile2 install &&  
mv /usr/bin/kill /bin
```

Command explanations

`sed "s/XConsole/#XConsole/" Makefile > Makefile2`: This will comment out the `XConsole` variable in the `Makefile`. This utility can't be build yet because we don't have X installed yet.

Contents

The Procps package contains the `free`, `kill`, `oldps`, `ps`, `skill`, `snice`, `sysctl`, `tload`, `top`, `uptime`, `vmstat`, `w` and `watch` programs.

Description

free

`free` displays the total amount of free and used physical and swap memory in the system, as well as the shared memory and buffers used by the kernel.

kill

`kill` sends signals to processes.

oldps and ps

ps gives a snapshot of the current processes.

skill

skill sends signals to process matching a criteria.

snice

snice changes the scheduling priority for process matching a criteria.

sysctl

sysctl modifies kernel parameters at runtime.

tload

tload prints a graph of the current system load average to the specified tty (or the tty of the tload process if none is specified).

top

top provides an ongoing look at processor activity in real time.

uptime

uptime gives a one line display of the following information: the current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

vmstat

vmstat reports information about processes, memory, paging, block IO, traps, and cpu activity.

w

w displays information about the users currently on the machine, and their processes.

watch

watch runs command repeatedly, displaying its output (the first screenfull).

Installing Psmisc

Installation of Psmisc

Edit the `Makefile` file and edit the `CFLAGS` variable if you want to add compiler optimizations to this package.

Install Psmisc by running the following commands:

```
sed "s/-ltermcap/-lncurses/" Makefile >Makefile2 &&  
make -f Makefile2 &&  
make install
```

Contents

The Psmisc package contains the `fuser`, `killall` and `pstree` programs.

Description

fuser

`fuser` displays the PIDs of processes using the specified files or file systems.

killall

`killall` sends a signal to all processes running any of the specified commands.

pstree

`pstree` shows running processes as a tree.

Installing Sed

Installation of Sed

Install Sed by running the following commands:

```
./configure --prefix=/usr --bindir=/bin &&  
make &&  
make install
```

Contents

The Sed package contains the sed program.

Description

sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline).

Installing Shellutils

Installation of Sh-utils

Install Shellutils by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install &&  
cd /usr/bin &&  
mv date echo false pwd stty /bin &&  
mv su true uname hostname /bin
```

Contents

The Shellutils package contains the basename, chroot, date, dirname, echo, env, expr, factor, false, groups, hostid, hostname, id, logname, nice, nohup, pathchk, pinky, printenv, printf, pwd, seq, sleep, stty, su, tee, test, true, tty, uname, uptime, users, who, whoami and yes programs.

Description

basename

basename strips directory and suffixes from filenames.

chroot

chroot runs a command or interactive shell with special root directory.

date

date displays the current time in a specified format, or sets the system date.

dirname

dirname strips non-directory suffixes from file name.

echo

echo displays a line of text.

env

env runs a program in a modified environment.

expr

expr evaluates expressions.

factor

factor prints the prime factors of all specified integer numbers.

false

false always exits with a status code indicating failure.

groups

groups prints the groups a user is in.

hostid

hostid prints the numeric identifier (in hexadecimal) for the current host.

hostname

hostname sets or prints the name of the current host system

id

id prints the real and effective UIDs and GIDs of a user or the current user.

logname

logname prints the current user's login name.

nice

nice runs a program with modified scheduling priority.

nohup

nohup runs a command immune to hangups, with output to a non-tty

pathchk

pathchk checks whether file names are valid or portable.

pinky

pinky is a lightweight finger utility which retrieves information about a certain user

printenv

printenv prints all or part of the environment.

printf

printf formats and print data (the same as the printf C function).

pwd

pwd prints the name of the current/working directory

seq

seq prints numbers in a certain range with a certain increment.

sleep

sleep delays for a specified amount of time.

stty

stty changes and prints terminal line settings.

su

su runs a shell with substitute user and group IDs

tee

tee reads from standard input and write to standard output and files.

test

test checks file types and compares values.

true

True always exitx with a status code indicating success.

tty

tty prints the file name of the terminal connected to standard input.

uname

uname prints system information.

uptime

uptime tells how long the system has been running.

users

users prints the user names of users currently logged in to the current host.

who

who shows who is logged on.

whoami

whoami prints your effective userid.

yes

yes outputs a string repeatedly until killed.

Installing Shadowpwd

Installation of Shadow Password Suite

Install the Shadow Password Suite by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install &&  
cd etc &&  
cp limits login.access \  
    login.defs.linux shells suauth /etc &&  
mv /etc/login.defs.linux /etc/login.defs
```

Command explanations

cp limits login.access and others: These files were not installed during the installation of the package so we copy them manually as those files are used to configure authentication details on your system.

Contents

The Shadow Password Suite contains the chage, chfn, chsh, expiry, faillog, gpasswd, lastlog, login, newgrp, passwd, sg, su, chpasswd, dpasswd, groupadd, groupdel, groupmod, grpck, grpconv, grpunconv, logoutd, mkpasswd, newusers, pwck, pwconv, pwunconv, useradd, userdel, usermod and vipw programs.

Description

chage

chage changes the number of days between password changes and the date of the last password change.

chfn

chfn changes user fullname, office number, office extension, and home phone number information for a user's account.

chsh

chsh changes the user login shell.

expiry

It's currently unknown what this program is for.

faillog

faillog formats the contents of the failure log, `/var/log/faillog`, and maintains failure counts and limits.

gpasswd

gpasswd is used to administer the `/etc/group` file

lastlog

lastlog formats and prints the contents of the last login log, `/var/log/lastlog`. The login-name, port, and last login time will be printed.

login

login is used to establish a new session with the system.

newgrp

newgrp is used to change the current group ID during a login session.

passwd

passwd changes passwords for user and group accounts.

sg

sg executes command as a different group ID.

su

Change the effective user id and group id to that of a user. This replaces the su programs that's installed from the Shellutils package.

chpasswd

chpasswd reads a file of user name and password pairs from standard input and uses this information to update a group of existing users.

dpasswd

dpasswd adds, deletes, and updates dialup passwords for user login shells.

groupadd

The groupadd command creates a new group account using the values specified on the command line and the default values from the system.

groupdel

The groupdel command modifies the system account files, deleting all entries that refer to group.

groupmod

The groupmod command modifies the system account files to reflect the changes that are specified on the command line.

grpck

grpck verifies the integrity of the system authentication information.

grpconv

grpconv converts to shadow group files from normal group files.

grpunconv

grpunconv converts from shadow group files to normal group files.

logoutd

logoutd enforces the login time and port restrictions specified in /etc/porttime.

mkpasswd

mkpasswd reads a file in the format given by the flags and converts it to the corresponding database file format.

newusers

newusers reads a file of user name and cleartext password pairs and uses this information to update a group of existing users or to create new users.

pwck

pwck verifies the integrity of the system authentication information.

pwconv

pwconv converts to shadow passwd files from normal passwd files.

pwunconv

pwunconv converts from shadow passwd files to normal files.

useradd

useradd creates a new user or update default new user information.

userdel

userdel modifies the system account files, deleting all entries that refer to a specified login name.

usermod

usermod modifies the system account files to reflect the changes that are specified on the command line.

vipw and vigr

vipw and vigr will edit the files /etc/passwd and /etc/group, respectively. With the `-s` flag, they will edit the shadow versions of those files, /etc/shadow and /etc/gshadow, respectively.

Installing Sysklogd

Installation of Sysklogd

Edit the `Makefile` file and edit the `CFLAGS` variable if you want to add compiler optimizations to this package.

Install Sysklogd by running the following commands:

```
make &&  
make install
```

Contents

The Sysklogd package contains the `klogd` and `syslogd` programs.

Description

klogd

`klogd` is a system daemon which intercepts and logs Linux kernel messages.

syslogd

`Syslogd` provides a kind of logging that many modern programs use. Every logged message contains at least a time and a hostname field, normally a program name field, too, but that depends on how trustworthy the logging program is.

Installing Sysvinit

Installation of Sysvinit

Edit the `src/Makefile` file and edit the `CFLAGS` variable if you want to add compiler optimizations to this package.

Install Sysvinit by running the following commands:

```
cd src &&  
make &&  
make install
```

Contents

The Sysvinit package contains the `pidof`, `last`, `lastb`, `mesg`, `utmpdump`, `wall`, `halt`, `init`, `killall5`, `poweroff`, `reboot`, `runlevel`, `shutdown`, `sulogin` and `telinit` programs.

Description

pidof

`Pidof` finds the process id's (pids) of the named programs and prints those id's on standard output.

last

`last` searches back through the file `/var/log/wtmp` (or the file designated by the `-f` flag) and displays a list of all users logged in (and out) since that file was created.

lastb

`lastb` is the same as `last`, except that by default it shows a log of the file `/var/log/btmp`, which contains all the bad login attempts.

mesg

`Mesg` controls the access to your terminal by others. It's typically used to allow or disallow other users to

write to your terminal.

utmpdump

utmpdumps prints the content of a file (usually `/var/run/utmp`) on standard output in a user friendly format.

wall

Wall sends a message to everybody logged in with their `mesg` permission set to `yes`.

halt

Halt notes that the system is being brought down in the file `/var/log/wtmp`, and then either tells the kernel to halt, reboot or `poweroff` the system. If `halt` or `reboot` is called when the system is not in runlevel 0 or 6, shutdown will be invoked instead (with the flag `-h` or `-r`).

init

Init is the parent of all processes. Its primary role is to create processes from a script stored in the file `/etc/inittab`. This file usually has entries which cause `init` to spawn `gettys` on each line that users can log in. It also controls autonomous processes required by any particular system.

killall5

`killall5` is the SystemV `killall` command. It sends a signal to all processes except the processes in its own session, so it won't kill the shell that is running the script it was called from.

poweroff

`poweroff` is equivalent to `shutdown -h -p now`. It halts the computer and switches off the computer (when using an APM compliant BIOS and APM is enabled in the kernel).

reboot

`reboot` is equivalent to `shutdown -r now`. It reboots the computer.

runlevel

Runlevel reads the system `utmp` file (typically `/var/run/utmp`) to locate the runlevel record, and then prints

the previous and current system runlevel on its standard output, separated by a single space.

shutdown

shutdown brings the system down in a secure way. All logged-in users are notified that the system is going down, and login is blocked.

sulogin

sulogin is invoked by init when the system goes into single user mode (this is done through an entry in /etc/inittab). Init also tries to execute sulogin when it is passed the -b flag from the bootmonitor (eg, LILO).

telinit

telinit sends appropriate signals to init, telling it which runlevel to change to.

Installing Tar

Installation of Tar

If you want to apply the Bzip2 tar patch which gives you the `-y` option to tar so you can use bzip2 files with tar, first download the patch from <http://sourceware.cygnum.com/bzip2/> and apply it by running the following command within the src directory under the tar-1.13 directory:

```
cd src &&  
patch -i ../../gnutarpatch.txt &&  
cd ..
```

Install Tar by running the following commands from the toplevel directory:

```
./configure --prefix=/usr &&  
make &&  
make install &&  
mv /usr/bin/tar /bin
```

Contents

The tar package contains the tar and rmt programs.

Description

tar

tar is an archiving program designed to store and extract files from an archive file known as a tarfile.

rmt

rmt is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection.

Installing Textutils

Installation of Textutils

Install Textutils by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install &&  
mv /usr/bin/cat /bin
```

Contents

The Textutils package contains the cat, cksum, comm, split, cut, expand, fmt, fold, head, join, md5sum, nl, od, paste, pr, ptx, sort, split, sum, tac, tail, tr, tsort, unexpand, uniq and wc programs.

Description

cat

cat concatenates file(s) or standard input to standard output.

cksum

cksum prints CRC checksum and byte counts of each specified file.

comm

comm compares two sorted files line by line.

csplit

csplit outputs pieces of a file separated by (a) pattern(s) to files xx01, xx02, ..., and outputs byte counts of each piece to standard output.

cut

cut prints selected parts of lines from specified files to standard output.

expand

expand converts tabs in files to spaces, writing to standard output.

fmt

fmt reformats each paragraph in the specified file(s), writing to standard output.

fold

fold wraps input lines in each specified file (standard input by default), writing to standard output.

head

Print first xx (10 by default) lines of each specified file to standard output.

join

join joins lines of two files on a common field.

md5sum

md5sum prints or checks MD5 checksums.

nl

nl writes each specified file to standard output, with line numbers added.

od

od writes an unambiguous representation, octal bytes by default, of a specified file to standard output.

paste

paste writes lines consisting of the sequentially corresponding lines from each specified file, separated by TABs, to standard output.

pr

pr paginates or columnates files for printing.

ptx

ptx produces a permuted index of file contents.

sort

sort writes sorted concatenation of files to standard output.

split

split outputs fixed-size pieces of an input file to PREFIXaa, PREFIXab, ...

sum

sum prints checksum and block counts for each specified file.

tac

tac writes each specified file to standard output, last line first.

tail

tail print the last xx (10 by default) lines of each specified file to standard output.

tr

tr translates, squeezes, and/or deletes characters from standard input, writing to standard output.

tsort

tsort writes totally ordered lists consistent with the partial ordering in specified files.

unexpand

unexpand converts spaces in each file to tabs, writing to standard output.

uniq

uniq discards all but one of successive identical lines from files or standard input and writes to files or standard output.

wc

wc prints line, word, and byte counts for each specified file, and a total line if more than one file is specified.

Installing Utillinux

Installation of Util-Linux

Before we can install the package we have to edit the MCONFIG file, find and modify the following variables as follows:

```
HAVE_SLN=yes  
HAVE_TSORT=yes
```

Now find the following lines in the MCONFIG file:

```
# Different optimizations for different cpus.  
ifeq "$(ARCH)" "intel"  
  ifeq "$(CPU)" "i386"  
    OPT=      -pipe -O2 -m386 -fomit-frame-pointer  
  else  
    OPT=      -pipe -O2 -m486 -fomit-frame-pointer  
  endif  
else  
<cut the rest of the lines in this optimizations section>
```

Modify the proper OPT variable to include the `-march` option. If you modify the first or second OPT variable, replace `-m486` with the `-march` variable and value you wish to use like `-march=i686`.

Install Util-Linux by running the following commands:

```
./configure &&  
make &&  
make install
```

Command explanations

HAVE_SLN=yes: We don't build this program because it already was installed by Glibc.

HAVE_TSORT=yes: We don't build this program either because it already was installed by Textutils.

Contents

The Util-linux package contains the arch, dmesg, kill, more, mount, umount, agetty, blockdev, cfdisk, ctrlaltdel, elvtune, fdisk, fsck.minix, hwclock, kbdrate, losetup, mkfs, mkfs.bfs, mkfs.minix, mkswap, sfdisk, swapoff, swapon, cal, chkdupexe, col, colcrt, colrm, column, cytune, ddate, fdformat, getopt, hexdump, ipcrm, ipcs, logger, look, mcookie, namei, rename, renice, rev, script, setfdprm, setsid, setterm, ul, whereis, write, ramsize, rdev, readprofile, rootflags, swapdev, tunelp and vidmode programs.

Description

arch

arch prints the machine architecture.

dmesg

dmesg is used to examine or control the kernel ring buffer (boot messages from the kernel).

kill

kill sends a specified signal to the specified process.

more

more is a filter for paging through text one screenful at a time.

mount

mount mounts a filesystem from a device to a directory (mount point).

umount

umount unmounts a mounted filesystem.

agetty

agetty opens a tty port, prompts for a login name and invokes the /bin/login command.

blockdev

No description available.

cfdisk

cfdisk is an libncurses based disk partition table manipulator.

ctrlaltdel

ctrlaltdel sets the function of the CTRL+ALT+DEL key combination (hard or soft reset).

elvtune

elvtune allows to tune the I/O elevator per blockdevice queue basis.

fdisk

fdisk is a disk partition table manipulator.

fsck.minix

fsck.minix performs a consistency check for the Linux MINIX filesystem.

hwclock

hwclock queries and sets the hardware clock (Also called the RTC or BIOS clock).

kbdrate

kbdrate resets the keyboard repeat rate and delay time.

losetup

losetup sets up and controls loop devices.

mkfs

mkfs builds a Linux filesystem on a device, usually a harddisk partition.

mkfs.bfs

mkfs.bfs creates a SCO bfs file system on a device, usually a harddisk partition.

mkfs.minix

mkfs.minix creates a Linux MINIX filesystem on a device, usually a harddisk partition.

mkswap

mkswap sets up a Linux swap area on a device or in a file.

sfdisk

sfdisk is a disk partition table manipulator.

swapoff

swapoff disables devices and files for paging and swapping.

swapon

swapon enables devices and files for paging and swapping.

cal

cal displays a simple calender.

chkdupexe

chkdupexe finds duplicate executables.

col

col filters reverse line feeds from input.

colcrt

colcrt filters nroff output for CRT previewing.

colrm

colrm removes columns from a file.

column

column columnates lists.

cytune

cytune queries and modifies the interruption threshold for the Cyclades driver.

ddate

ddate converts Gregorian dates to Discordian dates.

fdformat

fdformat low-level formats a floppy disk.

getopt

getops parses command options the same way as the getopt C command.

hexdump

hexdump displays specified files, or standard input, in a user specified format (ascii, decimal, hexadecimal, octal).

ipcrm

ipcrm removes a specified resource.

ipcs

ipcs provides information on ipc facilities.

logger

logger makes entries in the system log.

look

look displays lines beginning with a given string.

mcookie

mcookie generates magic cookies for xauth.

namei

namei follows a pathname until a terminal point is found.

rename

rename renames files.

renice

renice alters priority of running processes.

rev

rev reverses lines of a file.

script

script makes typescript of terminal session.

setfdprm

setfdprm sets user—provides floppy disk parameters.

setsid

setsid runs programs in a new session.

setterm

setterm sets terminal attributes.

ul

ul reads a file and translates occurrences of underscores to the sequence which indicates underlining for the terminal in use.

whereis

whereis locates a binary, source and manual page for a command.

write

write sends a message to another user.

ramsize

ramsize queries and sets RAM disk size.

rdev

rdev queries and sets image root device, swap device, RAM disk size, or video mode.

readprofile

readprofile reads kernel profiling information.

rootflags

rootflags queries and sets extra information used when mounting root.

swapdev

swapdev queries and sets swap device.

tunelp

tunelp sets various parameters for the lp device.

vidmode

vidmode queries and sets the video mode.

Installing Man–pages

Installation of Man–pages

Install Man–pages by running the following commands:

```
yes n|cp -avi man* /usr/share/man
```

Command explanations

yes n|cp -avi man* /usr/share/man: This copies all the files from all subdirectories to the /usr/share/man directory but when it asks to overwrite the files (that's what -i does – interactive, asks to overwrite files) we answer with "N".

Contents

The Man–pages package contains various manual pages that don't come with the packages.

Description

Examples of provided manual pages are the manual pages describing all the C and C++ functions, few important /dev/ files and more.

Removing old NSS library files

If you have copied the NSS Library files from your normal Linux system to the LFS system (because your normal system runs glibc-2.0) it's time to remove them now by running:

```
rm /lib/libnss*.so.1 /lib/libnss*2.0*
```

Configuring essential software

Now that all software is installed, all that we need to do to get a few programs running properly is to create their configuration files.

Configuring Vim

By default Vim runs in vi compatible mode. Some people might like this, but I have a high preference to run vim in vim mode (else I wouldn't have included Vim in this book but the original Vi). Create the `/root/.vimrc` by running the following:

```
cat > /root/.vimrc << "EOF"
" Begin /root/.vimrc

set nocompatible
set bs=2

" End /root/.vimrc
EOF
```

Configuring Glibc

We need to create the `/etc/nsswitch.conf` file. Although glibc should provide defaults when this file is missing or corrupt, it's defaults don't work well with networking which will be dealt with in a later chapter. Also, our timezone needs to be setup.

Create a new file `/etc/nsswitch.conf` by running the following:

```
cat > /etc/nsswitch.conf << "EOF"
# Begin /etc/nsswitch.conf

passwd: files
group: files
shadow: files

publickey: files

hosts: files dns
networks: files

protocols: db files
```

```
services: db files
ethers: db files
rpc: db files
```

```
netgroup: db files
```

```
# End /etc/nsswitch.conf
EOF
```

Run the **tzselect** script and answer the questions regarding your timezone. When you're done, the script will give you the location of the timezone file you need.

Create the `/etc/localtime` symlink by running:

```
cd /etc &&
rm localtime &&
ln -s ../usr/share/zoneinfo/<tzselect's output> localtime
```

tzselect's output can be something like *EST5EDT* or *Canada/Eastern*.

The symlink you would create with that information would be:

```
ln -s ../usr/share/zoneinfo/EST5EDT localtime
```

Or:

```
ln -s ../usr/share/zoneinfo/Canada/Eastern localtime
```

Configuring Dynamic Loader

By default the dynamic loader searches a few default paths for dynamic libraries, so there normally isn't a need for the `/etc/ld.so.conf` file unless you have extra directories in which you want the system to search for paths. The `/usr/local/lib` directory isn't searched through for dynamic libraries by default, so we want to add this path so when you install software you won't be suprised by them not running for some reason.

Create a new file `/etc/ld.so.conf` by running the following:

```
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf

/lib
/usr/lib
/usr/local/lib

# End /etc/ld.so.conf
EOF
```

Although it's not necessary to add the `/lib` and `/usr/lib` directories it doesn't hurt. This way you see right away what's being searched and don't have to remember the default search paths if you don't want to.

Configuring Lilo

We're not going to create lilo's configuration file from scratch, but we'll use the file from your normal Linux system. This file is different on every machine and thus I can't create it here. Since you would want to have the same options regarding lilo as you have when you're using your normal Linux system you would create the file exactly as it is on the normal system.

Copy the Lilo configuration file and kernel images that Lilo uses by running the following commands from a shell on your normal Linux system. Don't execute these commands from your chroot'ed shell.

```
cp /etc/lilo.conf $LFS/etc
cp /boot/<kernel images> $LFS/boot
```

Before you can execute the second command you need to know the names of the kernel images. You can't just copy all files from the `/boot` directory. The `/etc/lilo.conf` file contains the names of the kernel images you're using. Open the file and look for lines like this:

```
image=/boot/vmlinuz
```

Look for all *image* variables and their values represent the name and location of the image files. These files will usually be in `/boot` but they might be in other directories as well, depending on your distribution's conventions.

Configuring Syslogd

Create a new file `/etc/syslog.conf` by running the following:

```
cat > /etc/syslog.conf << "EOF"
# Begin /etc/syslog.conf

auth,authpriv.* /var/log/auth.log
*.*;auth,authpriv.none /var/log/syslog
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
```

Configuring Shadow Password Suite

This package contains the utilities to modify user's passwords, add new users/groups, delete users/groups and more. I'm not going to explain to you what 'password shadowing' means. You can read all about that in the `doc/HOWTO` file within the unpacked shadow password suite's source tree. There's one thing you should keep in mind, if you decide to use shadow support, that programs that need to verify passwords (examples are `xdm`, `ftp` daemons, `pop3` daemons, etc) need to be 'shadow-compliant', eg. they need to be able to work with shadow'ed passwords.

Shadow'ed passwords are not enabled by default. Simply installing the shadow password suite does not enable shadow'ed passwords.

Now is a very good moment to read chapter 5 of the `doc/HOWTO` file. You can read how you can enable shadow'ed passwords, how to test whether shadowing works and if not, how to disable it again.

The documentation mentions something about the creation of `npasswd` and `nshadow` after you run `pwconv`. This is an error in the documentation. Those two files will be created. After you run `pwconv`, `/etc/passwd` will no longer contain the passwords and `/etc/shadow` will. You don't need to rename the `npasswd` and `nshadow` files yourself.

Configuring Sysvinit

Create a new file `/etc/inittab` by running the following:

```

cat > /etc/inittab << "EOF"
# Begin /etc/inittab

id:3:initdefault:

si::sysinit:/etc/init.d/rcS

l0:0:wait:/etc/init.d/rc 0
l1:S1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

f1:0:respawn:/sbin/sulogin
f2:6:respawn:/sbin/sulogin

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S1:respawn:/sbin/sulogin
1:2345:respawn:/sbin/agetty 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

```

Creating the `/var/run/utmp`, `/var/log/wtmp` and `/var/log/btmp` files

Programs like `login`, `shutdown`, `uptime` and others want to read from and write to the `/var/run/utmp`, `/var/log/btmp` and `/var/log/wtmp`. These files contain information about who is currently logged in. It also contains information on when the computer was last booted and shutdown and a record of the bad login attempts.

Create these files with their proper permissions by running the following commands:

```

touch /var/run/utmp /var/log/wtmp /var/log/btmp
/var/log/lastlog &&
chmod 644 /var/run/utmp /var/log/wtmp /var/log/btmp

```

```
/var/log/lastlog
```

Creating root password

Choose a password for user root and create it by running the following command:

```
passwd root
```

Chapter 7. Creating system boot scripts

Introduction

This chapter will create the necessary scripts that are run at boottime. These scripts perform tasks such as remounting the root file system mounted read-only by the kernel into read-write mode, activating the swap partition(s), running a check on the root file system to make sure it's intact and starting the daemons that the system uses.

If you rather download these scripts than create them yourself you can get them from

<http://download.linuxfromscratch.org/bootscripts/> or <ftp://download.linuxfromscratch.org/bootscritps/>

Creating directories

We need to start by creating a few extra directories that are used by the boot scripts. Create these directories by running:

```
cd /etc &&  
mkdir sysconfig rc0.d rc1.d rc2.d rc3.d &&  
mkdir rc4.d rc5.d rc6.d init.d rcS.d &&  
cd init.d
```

Creating the rc script

The first main bootscript is the `/etc/init.d/rc` script. Create a new file `/etc/init.d/rc` containing the following:

```
cat > rc << "EOF"
#!/bin/sh
# Begin /etc/init.d/rc
#
# By Jason Pearce – jason.pearce@linux.org
# Modified by Gerard Beekmans – gerard@linuxfromscratch.org
# print_error_msg based on ideas by Simon Perreault – nomis80@yahoo.com

source /etc/init.d/functions

print_error_msg()
{
    echo
    $FAILURE
    echo -n "You should not read this error message. It means "
    echo "that an unforeseen error "
    echo -n "took place and subscript $i exited with "
    echo "a return value "
    echo -n "of $error_value for an unknown reason. If you're able "
    echo "to trace this error down "
    echo -n "to a bug in one of the files provided by this book, "
    echo "please be so kind to "
    echo -n "inform us at lfs-discuss@linuxfromscratch.org"
    $NORMAL
    echo
}

# Un-comment the following for debugging.
# debug=echo

#
# Start script or program.
#
startup() {
case "$1" in
    *.sh)
        $debug sh "$@"
        ;;
    *)
        $debug "$@"
        ;;
esac
}
```

```

esac
}

# Ignore CTRL-C only in this shell, so we can interrupt subprocesses.
trap ":" INT QUIT TSTP

# Set onlcr to avoid staircase effect.
stty onlcr 0>&1

# Now find out what the current and what the previous runlevel are.
runlevel=$RUNLEVEL
# Get first argument. Set new runlevel to this argument.

[ "$1" != "" ] && runlevel=$1
if [ "$runlevel" = "" ]
then
    echo "Usage: $0 <runlevel>" >&2
    exit 1
fi

previous=$PREVLEVEL
[ "$previous" = "" ] && previous=N

export runlevel previous

# Is there an rc directory for this new runlevel?

if [ -d /etc/rc$runlevel.d ]
then
    # First, run the KILL scripts for this runlevel.
    if [ $previous != N ]
    then
        for i in /etc/rc$runlevel.d/K*
        do
            [ ! -f $i ] && continue

            suffix=${i#/etc/rc$runlevel.d/K[0-9][0-9]}
            previous_start=/etc/rc$previous.d/S[0-9][0-9]$suffix
            sysinit_start=/etc/rcS.d/S[0-9][0-9]$suffix

            # Stop the service if there is a start script
            # in the previous run level.
            [ ! -f $previous_start ] &&
            [ ! -f $sysinit_start ] && continue

            startup $i stop
            error_value=$?
            if [ $error_value != 0 ]
            then
                print_error_msg
            fi
        done
    fi

```

```

done
fi

# Now run the START scripts for this runlevel.
for i in /etc/rc$runlevel.d/S*
do
    [ ! -f $i ] && continue

    if [ $previous != N ]
    then
        # Find start script in previous runlevel and
        # stop script in this runlevel.
        suffix=${i#/etc/rc$runlevel.d/S[0-9][0-9]}
        stop=/etc/rc$runlevel.d/K[0-9][0-9]$suffix
        previous_start=/etc/rc$previous.d/S[0-9][0-9]$suffix

        # If there is a start script in the previous
        # level
        # and _no_ stop script in this level, we don't
        # have to re-start the service.
        [ -f $previous_start ] && [ ! -f $stop ] &&
        continue
    fi

    case "$runlevel" in
        0|6)
            startup $i stop
            error_value=$?
            if [ $error_value != 0 ]
            then
                print_error_msg
            fi
            ;;
        *)
            startup $i start
            error_value=$?
            if [ $error_value != 0 ]
            then
                print_error_msg
            fi
            ;;
    esac
done
fi

# End /etc/init.d/rc
EOF

```

Creating the rcS script

The second main bootscript is the rcS script. Create a new file `/etc/init.d/rcS` containing the following:

```
cat > rcS << "EOF"
#!/bin/sh
# Begin /etc/init.d/rcS

runlevel=S
prevlevel=N
umask 022
export runlevel prevlevel

trap ":" INT QUIT TSTP

for i in /etc/rcS.d/S??*
do
    [ ! -f "$i" ] && continue;
    $i start
done

# End /etc/init.d/rcS
EOF
```

Creating the functions script

Create a new file `/etc/init.d/functions` containing the following:

```
cat > functions << "EOF"
#!/bin/sh
# Begin /etc/init.d/functions

COL=70
SET_COL="echo -en \\033[0;39m"
NORMAL="echo -en \\033[0;39m"
SUCCESS="echo -en \\033[1;32m"
FAILURE="echo -en \\033[1;31m"

evaluate_retval()
{
    if [ $? = 0 ]
    then
        print_status success
    else
        print_status failure
    fi
}

print_status()
{
    if [ $# = 0 ]
    then
        echo "Usage: print_status {success|failure}"
        exit 1
    fi

    case "$1" in
        success)
            $SET_COL
            echo -n "[ "
            $SUCCESS
            echo -n "OK"
            $NORMAL
            echo " ]"
            ;;
        failure)
            $SET_COL
            echo -n "[ "
            $FAILURE
            echo -n "FAILED"
            $NORMAL
            echo " ]"
```

```

        ;;
    esac
}

loadproc()
{
    if [ $# = 0 ]
    then
        echo "Usage: loadproc {program}"
        exit 1
    fi

    base=`usr/bin/basename $1`

    pidlist=`/bin/pidof -o $$ -o $PPID -o %PPID -x $base`

    pid=""

    for apid in $pidlist
    do
        if [ -d /proc/$apid ]
        then
            pid="$pid $apid"
        fi
    done

    if [ ! -n "$pid" ]
    then
        $*
        evaluate_retval
    else
        print_status failure
    fi
}

killproc()
{
    if [ $# = 0 ]
    then
        echo "Usage: killproc {program} [signal]"
        exit 1
    fi

    base=`usr/bin/basename $1`

    if [ "$2" != "" ]
    then
        killlevel=$2
    fi
}

```

```

else
    nolevel=1
fi

pidlist=`/bin/pidof -o $$ -o $PPID -o %PPID -x $base`

pid=""

for apid in $pidlist
do
    if [ -d /proc/$apid ]
    then
        pid="$pid $apid"
    fi
done

if [ -n "$pid" ]
then
    if [ "$nolevel" = 1 ]
    then
        /bin/kill -TERM $pid
        if ps h $pid >/dev/null 2>&1
        then
            /bin/kill -KILL $pid
        fi
        /bin/ps h $pid >/dev/null 2>&1
        if [ $? = 0 ]
        then
            print_status failure
        else
            /bin/rm -f /var/run/$base.pid
            print_status success
        fi
    else
        /bin/kill $skilllevel $pid
        /bin/ps h $pid >/dev/null 2>&1
        if [ $? = 0 ]
        then
            print_status failure
        else
            /bin/rm -f /var/run/$base.pid
            print_status success
        fi
    fi
else
    print_status failure
fi
}

reloadproc()
{

```

```

if [ $# = 0 ]
then
    echo "Usage: reloadproc {program} [signal]"
    exit 1
fi

base=`/usr/bin/basename $1`

if [ -n "$2" ]
then
    killlevel=-$2
else
    nolevel=1
fi

pidlist=`/bin/pidof -o $$ -o $PPID -o %PPID -x $base`

pid=""

for apid in $pidlist
do
    if [ -d /proc/$apid ]
    then
        pid="$pid $apid"
    fi
done

if [ -n "$pid" ]
then
    if [ "$nolevel" = 1 ]
    then
        /bin/kill -SIGHUP $pid
        evaluate_retval
    else
        /bin/kill -$killlevel $pid
        evaluate_retval
    fi
else
    print_status failure
fi
}

statusproc()
{
    if [ $# = 0 ]
    then
        echo "Usage: status {program}"
        return 1
    fi

    pid=`/bin/pidof -o $$ -o $PPID -o %PPID -x $1`

```

```
if [ -n "$pid" ]
then
    echo "$1 running with Process ID $pid"
    return 0
fi

if [ -f /var/run/$1.pid ]
then
    pid=`usr/bin/head -1 /var/run/$1.pid`
    if [ -n "$pid" ]
    then
        echo "$1 not running but /var/run/$1.pid exists"
        return 1
    fi
else
    echo "$1 is not running"
fi

}

# End /etc/init.d/functions
EOF
```

Creating the checkfs script

Create a new file `/etc/init.d/checkfs` containing the following:

```
cat > checkfs << "EOF"
#!/bin/sh
# Begin /etc/init.d/checkfs

source /etc/init.d/functions

echo -n "Activating swap..."
/sbin/swapon -a
evaluate_retval

if [ -f /fastboot ]
then
    echo "Fast boot, no file system check"
else
    /bin/mount -n -o remount,ro /
    if [ $? = 0 ]
    then
        if [ -f /forcefsck ]
        then
            echo -n "/forcefsck exists, forcing "
            echo "file system check"
            force="-f"
        else
            force=""
        fi

        echo "Checking file systems..."
        /sbin/fsck $force -a -A -C -T

        if [ $? -gt 1 ]
        then
            $FAILURE
            echo
            echo -n "fsck failed. Please repair your file "
            echo "systems manually by running /sbin/fsck"
            echo "without the -a option"
            echo
            echo -n "Please note that the root file system "
            echo "is currently mounted in read-only mode."
            echo
            echo -n "I will start slogin now. When you "
            echo "logout I will reboot your system."
            echo
            $NORMAL
        fi
    fi
fi
```

```
    /sbin/sulogin
    /sbin/reboot -f
else
    print_status success
fi

else
    echo -n "Cannot check root file system because it "
    echo "could not be mounted in read-only mode."
fi
fi

# End /etc/init.d/checkfs
EOF
```

Creating the halt script

Create a new file `/etc/init.d/halt` containing the following:

```
cat > halt << "EOF"
#!/bin/sh
# Begin /etc/init.d/halt

/sbin/halt -d -f -i -p

# End /etc/init.d/halt
EOF
```

Creating the loadkeys script

You only need to create this script if you don't have a default 101 keys US keyboard layout. Create a new file `/etc/init.d/loadkeys` containing the following:

```
cat > loadkeys << "EOF"
#!/bin/sh
# Begin /etc/init.d/loadkeys

source /etc/init.d/functions

echo -n "Loading keymap..."
/usr/bin/loadkeys -d >/dev/null
evaluate_retval

# End /etc/init.d/loadkeys
EOF
```

Creating the mountfs script

Create a new file `/etc/init.d/mountfs` containing the following:

```
cat > mountfs << "EOF"
#!/bin/sh
# Begin /etc/init.d/mountfs

source /etc/init.d/functions

echo -n "Remounting root file system in read-write mode..."
/bin/mount -n -o remount,rw /
evaluate_retval

echo > /etc/mtab
/bin/mount -f -o remount,rw /

/bin/rm -f /fastboot /forcefsck

echo -n "Mounting other file systems..."
/bin/mount -a
evaluate_retval

# End /etc/init.d/mountfs
EOF
```

Creating the reboot script

Create a new file `/etc/init.d/reboot` containing the following:

```
cat > reboot << "EOF"
#!/bin/sh
# Begin /etc/init.d/reboot

echo "System reboot in progress..."

/sbin/reboot -d -f -i

# End /etc/init.d/reboot
EOF
```

Creating the sendsignals script

Create a new file `/etc/init.d/sendsignals` containing the following:

```
cat > sendsignals << "EOF"
#!/bin/sh
# Begin /etc/init.d/sendsignals

source /etc/init.d/functions

echo -n "Sending all processes the TERM signal..."
/sbin/killall5 -15
evaluate_retval

echo -n "Sending all processes the KILL signal..."
/sbin/killall5 -9
evaluate_retval

# End /etc/init.d/sendsignals
EOF
```

Creating the setclock script

The following script is only for real use when your hardware clock (also known as BIOS or CMOS clock) isn't set to GMT time. The recommended setup is setting your hardware clock to GMT and have the time converted to localtime using the `/etc/localtime` symbolic link. But if you run an OS that doesn't understand a clock set to GMT (most notable are Microsoft OS'es) you might want to set your clock to localtime so that the time is properly displayed on those OS'es. This script will reset the kernel time to the hardware clock without converting the time using the `/etc/localtime` symlink.

If you want to use this script on your system even if you have your hardware clock set to GMT, then change the UTC variable below to the value of `1`.

```
cat > setclock << "EOF"
#!/bin/sh
# Begin /etc/init.d/setclock

source /etc/init.d/functions
source /etc/sysconfig/clock

CLOCKPARAMS="--hctosys"

case "$UTC" in
    yes|true|1)
        CLOCKPARAMS="$CLOCKPARAMS -u"
        ;;
esac

echo -n "Setting clock..."
/sbin/hwclock $CLOCKPARAMS
evaluate_retval

# End /etc/init.d/setclock
EOF
```

Creating the `/etc/sysconfig/clock` file

Create a new file `/etc/sysconfig/clock` by running the following:

```
cat > /etc/sysconfig/clock << "EOF"
# Begin /etc/sysconfig/clock

UTC=1
```

```
# End /etc/sysconfig/clock  
EOF
```

If your hardware clock (also known as BIOS or CMOS clock) is not set to GMT time, than set the UTC variable in the `/etc/sysconfig/clock` file to the value `0` (zero).

Creating the syslogd script

Create a new file `/etc/init.d/syslogd` containing the following:

```
cat > syslogd << "EOF"
#!/bin/sh
# Begin /etc/init.d/syslogd

source /etc/init.d/functions

case "$1" in
    start)
        echo -n "Starting system log daemon..."
        loadproc /usr/sbin/syslogd -m 0

        echo -n "Starting kernel log daemon..."
        loadproc /usr/sbin/klogd
        ;;

    stop)
        echo -n "Stopping kernel log daemon..."
        killproc klogd

        echo -n "Stopping system log daemon..."
        killproc syslogd
        ;;

    reload)
        echo -n "Reloading system log daemon configuration file..."
        reloadproc syslogd 1
        ;;

    restart)
        $0 stop
        /usr/bin/sleep 1
        $0 start
        ;;

    status)
        statusproc /usr/sbin/syslogd
        statusproc /usr/sbin/klogd
        ;;

    *)
        echo "Usage: $0 {start|stop|reload|restart|status}"
        exit 1
        ;;
endcase
```

```
esac
```

```
# End /etc/init.d/syslogd
```

```
EOF
```

Creating the template script

Create a new file `/etc/init.d/template` containing the following:

```
cat > template << "EOF"
#!/bin/sh
# Begin /etc/init.d/

source /etc/init.d/functions

case "$1" in
    start)
        echo -n "Starting ..."
        loadproc
        ;;

    stop)
        echo -n "Stopping ..."
        killproc
        ;;

    reload)
        echo -n "Reloading ..."
        reloadproc
        ;;

    restart)
        $0 stop
        /usr/bin/sleep 1
        $0 start
        ;;

    status)
        statusproc
        ;;

    *)
        echo "Usage: $0 {start|stop|reload|restart|status}"
        exit 1
        ;;

esac

# End /etc/init.d/
EOF
```

Creating the umountfs script

Create a new file `/etc/init.d/umountfs` containing the following:

```
cat > umountfs << "EOF"
#!/bin/sh
# Begin /etc/init.d/umountfs

source /etc/init.d/functions

echo -n "Deactivating swap..."
/sbin/swapoff -a
evaluate_retval

echo -n "Unmounting file systems..."
/bin/umount -a -r
evaluate_retval

# End /etc/init.d/umountfs
EOF
```

Setting up symlinks and permissions

Give these files the proper permissions and create the necessary symlinks by running the following commands. If you did not create the loadkeys and setclock scripts, make sure you don't type them in the commands below.

```
cd /etc/init.d &&
chmod 754 rc rcS functions checkfs halt loadkeys mountfs
reboot &&
chmod 754 sendsignals setclock sysklogd template umountfs &&
cd ../rc0.d &&
ln -s ../init.d/sysklogd K90sysklogd &&
ln -s ../init.d/sendsignals S80sendsignals &&
ln -s ../init.d/umountfs S90umountfs &&
ln -s ../init.d/halt S99halt &&
cd ../rc6.d &&
ln -s ../init.d/sysklogd K90sysklogd &&
ln -s ../init.d/sendsignals S80sendsignals &&
ln -s ../init.d/umountfs S90umountfs &&
ln -s ../init.d/reboot S99reboot &&
cd ../rcS.d &&
ln -s ../init.d/checkfs S05checkfs &&
ln -s ../init.d/mountfs S10mountfs &&
ln -s ../init.d/setclock S20setclock &&
ln -s ../init.d/loadkeys S30loadkeys &&
cd ../rc1.d &&
ln -s ../init.d/sysklogd K90sysklogd &&
cd ../rc2.d &&
ln -s ../init.d/sysklogd S03sysklogd &&
cd ../rc3.d &&
ln -s ../init.d/sysklogd S03sysklogd &&
cd ../rc4.d &&
ln -s ../init.d/sysklogd S03sysklogd &&
cd ../rc5.d &&
ln -s ../init.d/sysklogd S03sysklogd
```

Creating the /etc/fstab file

In order for certain programs to be able to determine where certain partitions are supposed to be mounted by default, the /etc/fstab file is used. Create a new file /etc/fstab containing the following:

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab

/dev/<LFS-partition designation> / ext2 defaults 1 1
/dev/<swap-partition designation> swap swap defaults 0 0
proc /proc proc defaults 0 0

# End /etc/fstab
EOF
```

Replace <LFS-partition designation> and <swap-partition designation> with the appropriate devices (/dev/hda5 and /dev/hda6 in my case).

Chapter 8. Making the LFS system bootable

Introduction

This chapter will make LFS bootable. This chapter deals with building a new kernel for our new LFS system and adding the proper entries to LILO so that you can select to boot the LFS system at the LILO: prompt.

Installing a kernel

A kernel is the heart of a Linux system. We could use the kernel image from our normal system, but we might as well compile a new kernel from the most recent kernel sources available.

Building the kernel involves a few steps: configuring it and compiling it. There are a few ways to configure the kernel. If you don't like the way this book does it, read the `README` file and find out what your other options are. Run the following commands to build the kernel:

```
cd /usr/src/linux &&  
make mrproper &&  
make menuconfig &&  
make dep &&  
make bzImage &&  
make modules &&  
make modules_install &&  
cp arch/i386/boot/bzImage /boot/lfskernel &&  
cp System.map /boot
```

Adding an entry to LILO

In order to being able to boot from this partition, we need to update our `/etc/lilo.conf` file. Add the following lines to `lilo.conf` by running:

```
cat >> /etc/lilo.conf << "EOF"
image=/boot/lfskernel
  label=lfs
  root=<partition>
  read-only
EOF
```

`<partition>` must be replaced by your partition's designation (which would be `/dev/hda5` in my case).

Now update the boot loader by running:

```
/sbin/lilo
```

Rebooting the system

Now that all software has been installed, bootscripts have been created, it's time for you to reboot your computer. Shutdown your system with `shutdown -r` now and reboot into LFS. After the reboot you will have a normal login prompt like you have on your normal Linux system (unless you use XDM or some sort of other Display Manger (like KDM – KDE's version of XDM)).

One thing remains to be done and that's setting up networking. After you rebooted and finished the next chapter of this book your LFS system is ready for use and you can do with it whatever you want.

Chapter 9. Setting up basic networking

Introduction

This chapter will setup basic networking. Although you might not be connected to a network, Linux software uses network functions anyway. We'll be installing at least the local loopback device and a network card as well if applicable. Also the proper bootscripts will be created so that networking will be enabled during boot time.

Installing network software

Installing Netkit-base

Install Netkit-base by running the following commands:

```
./configure --prefix=/usr &&  
make &&  
make install &&  
cd etc.sample &&  
cp services protocols /etc
```

There are other files in the `etc.sample` directory which might be of interest to you.

Installing Net-tools

Edit the `Makefile` file and edit the `CFLAGS` variable if you want to add compiler optimizations.

Install Net-tools by running the following commands:

```
make &&  
make install
```

You might have noticed that we don't use the compiler optimizations for this package. The reason is that overriding the `CFLAGS` variable causes compilation problems. You would have to edit the `Makefile` file and add the proper values to the `CFLAGS` variable and then compile the package. If you want to do that it's up to you. I don't think it's worth the trouble though. The programs in this package aren't that big that optimization would have any noticeable effect on the performance.

Creating network boot scripts

Creating the `/etc/init.d/localnet` bootscript

Create a new file `/etc/init.d/localnet` containing the following:

```
cat > /etc/init.d/localnet << "EOF"
#!/bin/sh
# Begin /etc/init.d/localnet

source /etc/init.d/functions
source /etc/sysconfig/network

case "$1" in
    start)
        echo -n "Bringing up the loopback interface..."
        /sbin/ifconfig lo 127.0.0.1
        evaluate_retval

        echo -n "Setting up hostname..."
        /bin/hostname $HOSTNAME
        evaluate_retval
        ;;

    stop)
        echo -n "Bringing down the loopback interface..."
        /sbin/ifconfig lo down
        evaluate_retval
        ;;

    *)
        echo "Usage: $0: {start|stop}"
        exit 1
        ;;
esac

# End /etc/init.d/localnet
EOF
```

Setting up permissions and symlink

Set the proper file permissions and create the necessary symlink by running the following commands:

```
cd /etc/init.d &&  
chmod 754 localnet &&  
cd ../rcS.d &&  
ln -s ../init.d/localnet S03localnet
```

Creating the `/etc/sysconfig/network` file

Create a new file `/etc/sysconfig/network` and put the `hostname` in it by running:

```
echo "HOSTNAME=lfs" > /etc/sysconfig/network
```

Replace "lfs" by the name you wish to call your computer. Please note that you should not enter the FQDN (Fully Qualified Domain Name) here. That information will be put in the `/etc/hosts` file later.

Creating the `/etc/hosts` file

If you want to configure a network card, you have to decide on the IP-address, FQDN and possible aliases for use in the `/etc/hosts` file. An example is:

```
<my-IP> myhost.mydomain.org aliases
```

Make sure the IP-address is in the private network IP-address range. Valid ranges are:

```
Class Networks  
A 10.0.0.0  
B 172.16.0.0 through 172.31.0.0  
C 192.168.0.0 through 192.168.255.0
```

A valid IP address could be 192.168.1.1. A valid FQDN for this IP could be `www.linuxfromscratch.org`

If you're not going to use a network card, you still need to come up with a FQDN. This is necessary for programs like Sendmail to operate correctly (in fact; Sendmail won't run when it can't determine the FQDN).

If you don't configure a network card, create a new file `/etc/hosts` by running:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (no network card version)

127.0.0.1 www.mydomain.com <value of HOSTNAME> localhost

# End /etc/hosts (no network card version)
EOF
```

If you do configure a network card, create a new file `/etc/hosts` containing:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1 localhost.localdomain localhost
192.168.1.1 www.mydomain.org <value of HOSTNAME>

# End /etc/hosts (network card version)
EOF
```

Of course, change the 192.168.1.1 and `www.mydomain.org` to your own liking (or requirements if you are assigned an IP-address by a network/system administrator and you plan on connecting this machine to that network).

Creating the `/etc/init.d/ethnet` script

This section only applies if you are going to configure a network card. If you're not, skip this section.

Create a new file `/etc/init.d/ethnet` containing the following:

```
cat > /etc/init.d/ethnet << "EOF"
#!/bin/sh
# Begin /etc/init.d/ethnet

source /etc/init.d/functions
source /etc/sysconfig/network

case "$1" in
    start)
```

```

for interface in `ls /etc/sysconfig/network-scripts/ifcfg* | grep -v ifcfg-lo`
do
    source $interface
    name=`usr/bin/basename $interface | sed 's/^ifcfg-//g`
    echo -n "Brining up the $name interface..."
    /sbin/ifconfig $name $IP broadcast $BROADCAST netmask $NETMASK
    evaluate_retval
done

#   echo -n "Setting up default route..."
#   /sbin/route add default gw $GATEWAY
#   evaluate_retval
    ;;

stop)

for interface in `ls /etc/sysconfig/network-scripts/ifcfg* | grep -v ifcfg-lo`
do
    source $interface
    name=`usr/bin/basename $interface | sed 's/^ifcfg-//g`
    echo -n "Brining down the $name interface..."
    /sbin/ifconfig $name down
    evaluate_retval
done
    ;;

*)
    echo "Usage: $0 {start|stop}"
    exit 1
    ;;
esac

# End /etc/init.d/ethnet
EOF

```

Adding default gateway to /etc/sysconfig/network

If you require a default gateway to be setup, uncomment the following three lines in the ethnet script:

```

#   echo -n "Setting up default route..."
#   /sbin/route add default gw $GATEWAY
#   evaluate_retval

```

And run the following command:

Adding default gateway to /etc/sysconfig/network

```
cat >> /etc/sysconfig/network << "EOF"  
GATEWAY=192.168.1.2  
EOF
```

Change GATEWAY to match your network setup.

Creating NIC configuration files

Which interfaces are brought up and down by the `ethnet` script depends on the files in the `/etc/sysconfig/network-scripts` directory. This directory should contain files in the form of `ifcfg-x` where `x` is the network interface to be brought up and down like `eth0`, `eth0:1`, `eth1:2`, and so forth.

First create the `network-scripts` directory by running:

```
mkdir /etc/sysconfig/network-scripts
```

Now, create new files in that directory containing the following. Example file names are `ifcfg-eth0`, `ifcfg-eth0:3` and `ifcfg-eth1:2`

```
IP=192.168.1.1  
NETMASK=255.255.255.0  
BROADCAST=192.168.1.255
```

Of course, change the values of those three variables in every file to match the proper setup. Usually `NETMASK` and `BROADCAST` will remain the same, just the `IP` will change per network interface.

Setting up permissions and symlink

Set the proper file permissions and create the necessary symlink by running the following commands:

```
cd /etc/init.d &&  
chmod 754 ethnet &&  
cd ../rc1.d &&
```

```
ln -s ../init.d/ethnet K80ethnet &&  
cd ../rc2.d &&  
ln -s ../init.d/ethnet K80ethnet &&  
cd ../rc3.d &&  
ln -s ../init.d/ethnet S10ethnet &&  
cd ../rc4.d &&  
ln -s ../init.d/ethnet S10ethnet &&  
cd ../rc5.d &&  
ln -s ../init.d/ethnet S10ethnet
```

III. Part III – Appendixes

Table of Contents

A. [Package descriptions](#)

B. [Resources](#)

C. [Official download locations](#)

Appendix A. Package descriptions

Introduction

This appendix describes the following aspect of each and every package that is installed in this book:

- What every package contains
- What every program from a package does

The packages are listed in the same order as they are installed in chapter 5 (Intel system) or chapter 11 (PPC systems).

Most information about these packages (especially the descriptions of it) come from the man pages from those packages. I'm not going to print the entire man page, just the core elements to make you understand what a program does. If you want to know full details on a program, I suggest you start by reading the complete man page in addition to this appendix.

You will also find that certain packages are documented more in depth than others. The reason is that I just happen to know more about certain packages than I know about others. If you have anything to add on the following descriptions, please don't hesitate to email me. This list is going to contain an in depth description of every package installed, but I can't do this on my own. I have had help from various people but more help is needed.

Please note that currently only what a package does is described and not why you need to install it. That will be added later.

Glibc

Contents

The Glibc package contains the GNU C Library.

Description

The C Library is a collection of commonly used functions in programs. This way a programmer doesn't need to create his own functions for every single task. The most common things like writing a string to your screen are already present and at the disposal of the programmer.

The C library (actually almost every library) come in two flavours: dynamic ones and static ones. In short when a program uses a static C library, the code from the C library will be copied into the executable file. When a program uses a dynamic library, that executable will not contain the code from the C library, but instead a routine that loads the functions from the library at the time the program is run. This means a significant decrease in the file size of a program. If you don't understand this concept, you better read the documentation that comes with the C Library as it is too complicated to explain here in one or two lines.

Linux kernel

Contents

The Linux kernel package contains the Linux kernel.

Description

The Linux kernel is at the core of every Linux system. It's what makes Linux tick. When you turn on your computer and boot a Linux system, the very first piece of Linux software that gets loaded is the kernel. The kernel initializes the system's hardware components such as serial ports, parallel ports, sound cards, network cards, IDE controllers, SCSI controllers and a lot more. In a nutshell the kernel makes the hardware available so that the software can run.

Ed

Contents

The Ed package contains the ed program.

Description

Ed is a line-oriented text editor. It is used to create, display, modify and otherwise manipulate text files.

Patch

Contents

The Patch package contains the patch program.

Description

The patch program modifies a file according to a patch file. A patch file usually is a list created by the diff program that contains instructions on how an original file needs to be modified. Patch is used a lot for source code patches since it saves time and space. Imagine you have a package that is 1MB in size. The next version of that package only has changes in two files of the first version. You can ship an entirely new package of 1MB or provide a patch file of 1KB which will update the first version to make it identical to the second version. So if you have downloaded the first version already, a patch file can save you a second large download.

GCC

Contents

The GCC package contains compilers, preprocessors and the GNU C++ Library.

Description

Compiler

A compiler translates source code in text format to a format that a computer understands. After a source code file is compiled into an object file, a linker will create an executable file from one or more of these compiler generated object files.

Pre-processor

A pre-processor pre-processes a source file, such as including the contents of header files into the source file. You generally don't do this yourself to save yourself a lot of time. You just insert a line like `#include <filename>`. The pre-processor file insert the contents of that file into the source file. That's one of the things a pre-processor does.

C++ Library

The C++ library is used by C++ programs. The C++ library contains functions that are frequently used in C++ programs. This way the programmer doesn't have to write certain functions (such as writing a string of text to the screen) from scratch every time he creates a program.

Bison

Contents

The Bison package contains the bison program.

Description

Bison is a parser generator, a replacement for YACC. YACC stands for Yet Another Compiler Compiler. What is Bison then? It is a program that generates a program that analyses the structure of a textfile. Instead of writing the actual program you specify how things should be connected and with those rules a program is constructed that analyses the textfile.

There are a lot of examples where structure is needed and one of them is the calculator.

Given the string :

$$1 + 2 * 3$$

You can easily come to the result 7. Why ? Because of the structure. You know how to interpret the string. The computer doesn't know that and Bison is a tool to help it understand by presenting the string in the following way to the compiler:

$$\begin{array}{c} + \\ / \backslash \\ * \ 1 \\ / \backslash \\ 2 \ 3 \end{array}$$

You start at the bottom of a tree and you come across the numbers 2 and 3 which are joined by the multiplication symbol, so the computer multiplies 2 and 3. The result of that multiplication is remembered and the next thing that the computer sees is the result of 2*3 and the number 1 which are joined by the add symbol. Adding 1 to the previous result makes 7. In calculating the most complex calculations can be broken down in this tree format and the computer just starts at the bottom and works its way up to the top and comes with the correct answer. Of course, Bison isn't only used for calculators alone.

Mawk

Contents

The Mawk package contains the mawk program.

Description

mawk

Mawk is an interpreter for the AWK Programming Language. The AWK language is useful for manipulation of data files, text retrieval and processing, and for prototyping and experimenting with algorithms.

Findutils

Contents

The Findutils package contains the find, locate, updatedb and xargs programs.

Description

Find

The find program searches for files in a directory hierarchy which match a certain criteria. If no criteria is given, it lists all files in the current directory and it's subdirectories.

Locate

Locate scans a database which contain all files and directories on a filesystem. This program lists the files and directories in this database matching a certain criteria. If you're looking for a file this program will scan the database and tell you exactly where the files you requested are located. This only makes sense if your locate database is fairly up-to-date else it will provide you with out-of-date information.

Updatedb

The updatedb program updates the locate database. It scans the entire file system (including other file system that are currently mounted unless you specify it not to) and puts every directory and file it finds into the database that's used by the locate program which retrieves this information. It's a good practice to update this database once a day so that you are ensured of a database that is up-to-date.

Xargs

The xargs command applies a command to a list of files. If you need to perform the same command on multiple files, you can create a file that contains all these files (one per line) and use xargs to perform that command on the list.

Ncurses

Contents

The Ncurses package contains the ncurses, panel, menu and form libraries. It also contains the tic, infocmp, clear, tput, toe and tset programs.

Description

The libraries

The libraries that make up the Ncurses library are used to display text (often in a fancy way) on your screen. An example where ncurses is used is in the kernel's "make menuconfig" process. The libraries contain routines to create panels, menu's, form and general text display routines.

Tic

Tic is the terminfo entry–description compiler. The program translates a terminfo file from source format into the binary format for use with the ncurses library routines. Terminfo files contain information about the capabilities of your terminal.

Infocmp

The infocmp program can be used to compare a binary terminfo entry with other terminfo entries, rewrite a terminfo description to take advantage of the use= terminfo field, or print out a terminfo description from the binary file (term) in a variety of formats (the opposite of what tic does).

clear

The clear program clears your screen if this is possible. It looks in the environment for the terminal type and then in the terminfo database to figure out how to clear the screen.

tput

The tput program uses the terminfo database to make the values of terminal–dependent capabilities and information available to the shell, to initialize or reset the terminal, or return the long name of the requested terminal type.

toe

The toe program lists all available terminal types by primary name with descriptions.

tset

The Tset program initializes terminals so they can be used, but it's not widely used anymore. It's provided for 4.4BSD compatibility.

Less

Contents

The Less package contains the less program

Description

The less program is a file pager (or text viewer). It displays the contents of a file with the ability to scroll. Less is an improvement on the common pager called "more". Less has the ability to scroll backwards through files as well and it doesn't need to read the entire file when it starts, which makes it faster when you are reading large files.

Groff

Contents

The Groff packages contains the addftinfo, afmtodit, eqn, grodvi, groff, grog, grohtml, grolj4, grops, grotty, hpftodit, indxbib, lkbib, lookbib, neqn, nroff, pfbtops, pic, psbb, refer, soelim, tbl, tfmtodit and troff programs.

Description

addftinfo

addftinfo reads a troff font file and adds some additional font–metric information that is used by the groff system.

afmtodit

afmtodit creates a font file for use with groff and grops.

eqn

eqn compiles descriptions of equations embedded within troff input files into commands that are understood by troff.

grodvi

grodvi is a driver for groff that produces TeX dvi format.

groff

groff is a front–end to the groff document formatting system. Normally it runs the troff program and a postprocessor appropriate for the selected device.

grog

grog reads files and guesses which of the groff options `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s`, and `-t` are required for printing files, and prints the groff command including those options on the standard output.

grohtml

grohtml translates the output of GNU troff to html

grolj4

grolj4 is a driver for groff that produces output in PCL5 format suitable for an HP Laserjet 4 printer.

grops

grops translates the output of GNU troff to PostScript.

grotty

grotty translates the output of GNU troff into a form suitable for typewriter-like devices.

hpftodit

hpftodit creates a font file for use with groff -Tlj4 from an HP tagged font metric file.

indxbib

indxbib makes an inverted index for the bibliographic databases a specified file for use with refer, lookbib, and lkbib.

lkbib

lkbib searches bibliographic databases for references that contain specified keys and prints any references found on the standard output.

lookbib

lookbib prints a prompt on the standard error (unless the standard input is not a terminal), reads from the standard input a line containing a set of keywords, searches the bibliographic databases in a specified file for references containing those keywords, prints any references found on the standard output, and repeats this process until the end of input.

neqn

It is currently not known what neqn is and what it does.

nroff

The nroff script emulates the nroff command using groff.

pfbtops

pfbtops translates a PostScript font in .pfb format to ASCII.

pic

pic compiles descriptions of pictures embedded within troff or TeX input files into commands that are understood by TeX or troff.

psbb

psbb reads a file which should be a PostScript document conforming to the Document Structuring conventions and looks for a %%BoundingBox comment.

refer

refer copies the contents of a file to the standard output, except that lines between .[and .] are interpreted as citations, and lines between .R1 and .R2 are interpreted as commands about how citations are to be processed.

soelim

soelim reads files and replaces lines of the form *.so file* by the contents of *file*.

tbl

tbl compiles descriptions of tables embedded within troff input files into commands that are understood by troff.

tfmtdit

tfmtdit creates a font file for use with **groff -Tdvi**

troff

troff is highly compatible with Unix troff. Usually it should be invoked using the groff command, which will also run preprocessors and postprocessors in the appropriate order and with the appropriate options.

Man

Contents

The Man package contains the man, apropos whatis and makewhatis programs.

Description

man

man formats and displays the on-line manual pages.

apropos

apropos searches a set of database files containing short descriptions of system commands for keywords and displays the result on the standard output.

whatis

whatis searches a set of database files containing short descriptions of system commands for keywords and displays the result on the standard output. Only complete word matches are displayed.

makewhatis

makewhatis reads all the manual pages contained in given sections of manpath or the preformatted pages contained in the given sections of catpath. For each page, it writes a line in the whatis database; each line consists of the name of the page and a short description, separated by a dash. The description is extracted using the content of the NAME section of the manual page.

Perl

Contents

The Perl package contains Perl – Practical Extraction and Report Language

Description

Perl combines the features and capabilities of C, awk, sed and sh into one powerful programming language.

M4

Contents

The M4 package contains the M4 processor

Description

M4 is a macro processor. It copies input to output expanding macros as it goes. Macros are either builtin or user-defined and can take any number of arguments. Besides just doing macro expansion m4 has builtin functions for including named files, running UNIX commands, doing integer arithmetic, manipulating text in various ways, recursion, etc. M4 can be used either as a front-end to a compiler or as a macro processor in its own right.

Texinfo

Contents

The Texinfo package contains the info, install-info, makeinfo, texi2dvi and texindex programs

Description

info

The info program reads Info documents, usually contained in your /usr/doc/info directory. Info documents are like man(ual) pages, but they tend to be more in depth than just explaining the options to a program.

install-info

The install-info program updates the info entries. When you run the info program a list with available topics (ie: available info documents) will be presented. The install-info program is used to maintain this list of available topics. If you decide to remove info files manually, you need to delete the topic in the index file as well. This program is used for that. It also works the other way around when you add info documents.

makeinfo

The makeinfo program translates Texinfo source documents into various formats. Available formats are: info files, plain text and HTML.

texi2dvi

The texi2dvi program prints Texinfo documents

texindex

The texindex program is used to sort Texinfo index files.

Autoconf

Contents

The Autoconf package contains the autoconf, autoheader, autoreconf, autoscan, autoupdate and ifnames programs

Description

autoconf

Autoconf is a tool for producing shell scripts that automatically configure software source code packages to adapt to many kinds of UNIX-like systems. The configuration scripts produced by Autoconf are independent of Autoconf when they are run, so their users do not need to have Autoconf.

autoheader

The autoheader program can create a template file of C #define statements for configure to use

autoreconf

If you have a lot of Autoconf-generated configure scripts, the autoreconf program can save you some work. It runs autoconf (and autoheader, where appropriate) repeatedly to remake the Autoconf configure scripts and configuration header templates in the directory tree rooted at the current directory.

autoscan

The autoscan program can help you create a configure.in file for a software package. autoscan examines source files in the directory tree rooted at a directory given as a command line argument, or the current directory if none is given. It searches the source files for common portability problems and creates a file configure.scan which is a preliminary configure.in for that package.

autoupdate

The autoupdate program updates a configure.in file that calls Autoconf macros by their old names to use the current macro names.

ifnames

ifnames can help when writing a configure.in for a software package. It prints the identifiers that the

package already uses in C preprocessor conditionals. If a package has already been set up to have some portability, this program can help you figure out what its configure needs to check for. It may help fill in some gaps in a configure.in generated by autoscan.

Automake

Contents

The Automake package contains the `aclocal` and `automake` programs

Description

`aclocal`

Automake includes a number of Autoconf macros which can be used in your package; some of them are actually required by Automake in certain situations. These macros must be defined in your `aclocal.m4`; otherwise they will not be seen by `autoconf`.

The `aclocal` program will automatically generate `aclocal.m4` files based on the contents of `configure.in`. This provides a convenient way to get Automake–provided macros, without having to search around. Also, the `aclocal` mechanism is extensible for use by other packages.

`automake`

To create all the `Makefile.in`'s for a package, run the `automake` program in the top level directory, with no arguments. `automake` will automatically find each appropriate `Makefile.am` (by scanning `configure.in`) and generate the corresponding `Makefile.in`.

Bash

Contents

The Bash package contains the bash program

Description

Bash is the Bourne–Again SHell, which is a widely used command interpreter on Unix systems. Bash is a program that reads from standard input, the keyboard. You type something and the program will evaluate what you have typed and do something with it, like running a program.

Flex

Contents

The Flex package contains the flex program

Description

Flex is a tool for generating programs which recognize patterns in text. Pattern recognition is very useful in many applications. You set up rules what to look for and flex will make a program that looks for those patterns. The reason people use flex is that it is much easier to set up rules for what to look for than to write the actual program that finds the text.

Binutils

Description

The Binutils package contains the ld, as, ar, nm, objcopy, objdump, ranlib, size, strings, strip, c++filt, addr2line and nlmconv programs

Description

ld

ld combines a number of object and archive files, relocates their data and ties up symbol references. Often the last step in building a new compiled program to run is a call to ld.

as

as is primarily intended to assemble the output of the GNU C compiler gcc for use by the linker ld.

ar

The ar program creates, modifies, and extracts from archives. An archive is a single file holding a collection of other files in a structure that makes it possible to retrieve the original individual files (called members of the archive).

nm

nm lists the symbols from object files.

objcopy

objcopy utility copies the contents of an object file to another. objcopy uses the GNU BFD Library to read and write the object files. It can write the destination object file in a format different from that of the source object file.

objdump

objdump displays information about one or more object files. The options control what particular information to display. This information is mostly useful to programmers who are working on the compilation tools, as opposed to programmers who just want their program to compile and work.

ranlib

ranlib generates an index to the contents of an archive, and stores it in the archive. The index lists each symbol defined by a member of an archive that is a relocatable object file.

size

size lists the section sizes —and the total size— for each of the object files objfile in its argument list. By default, one line of output is generated for each object file or each module in an archive.

strings

For each file given, strings prints the printable character sequences that are at least 4 characters long (or the number specified with an option to the program) and are followed by an unprintable character. By default, it only prints the strings from the initialized and loaded sections of object files; for other types of files, it prints the strings from the whole file.

strings is mainly useful for determining the contents of non-text files.

strip

strip discards all or specific symbols from object files. The list of object files may include archives. At least one object file must be given. strip modifies the files named in its argument, rather than writing modified copies under different names.

c++filt

The C++ language provides function overloading, which means that you can write many functions with the same name (providing each takes parameters of different types). All C++ function names are encoded into a low-level assembly label (this process is known as mangling). The c++filt program does the inverse mapping: it decodes (demangles) low-level names into user-level names so that the linker can keep these overloaded functions from clashing.

addr2line

addr2line translates program addresses into file names and line numbers. Given an address and an executable, it uses the debugging information in the executable to figure out which file name and line number are associated with a given address.

nlmconv

nlmconv converts relocatable object files into the NetWare Loadable Module files, optionally reading header files for NLM header information.

Bzip2

Contents

The Bzip2 packages contains the `bzip2`, `bunzip2`, `bzcat` and `bzip2recover` programs.

Description

Bzip2

`bzip2` compresses files using the Burrows–Wheeler block sorting text compression algorithm, and Huffman coding. Compression is generally considerably better than that achieved by more conventional LZ77/LZ78–based compressors, and approaches the performance of the PPM family of statistical compressors.

Bunzip2

`Bunzip2` decompresses files that are compressed with `bzip2`.

bzcat

`bzcat` (or `bzip2 -dc`) decompresses all specified files to the standard output.

bzip2recover

`bzip2recover` recovers data from damaged `bzip2` files.

Diffutils

Contents

The Diffutils package contains the `cmp`, `diff`, `diff3` and `sdiff` programs.

Description

`cmp` and `diff`

`cmp` and `diff` both compare two files and report their differences. Both programs have extra options which compare files in different situations.

`diff3`

The difference between `diff` and `diff3` is that `diff` compares 2 files, `diff3` compares 3 files.

`sdiff`

`sdiff` merges two two files and interactively outputs the results.

E2fsprogs

Contents

The e2fsprogs package contains the `chattr`, `lsattr`, `uuidgen`, `badblocks`, `debugfs`, `dumpe2fs`, `e2fsck`, `e2label`, `fsck`, `fsck.ext2`, `mke2fs`, `mkfs.ext2`, `mklost+found` and `tune2fs` programs.

Description

chattr

`chattr` changes the file attributes on a Linux second extended file system.

lsattr

`lsattr` lists the file attributes on a second extended file system.

uuidgen

The `uuidgen` program creates a new universally unique identifier (UUID) using the `libuuid` library. The new UUID can reasonably be considered unique among all UUIDs created on the local system, and among UUIDs created on other systems in the past and in the future.

badblocks

`badblocks` is used to search for bad blocks on a device (usually a disk partition).

debugfs

The `debugfs` program is a file system debugger. It can be used to examine and change the state of an ext2 file system.

dumpe2fs

`dumpe2fs` prints the super block and blocks group information for the filesystem present on a specified device.

e2fsck and fsck.ext2

e2fsck is used to check a Linux second extended file system. fsck.ext2 does the same as e2fsck.

e2label

e2label will display or change the filesystem label on the ext2 filesystem located on the specified device.

fsck

fsck is used to check and optionally repair a Linux file system.

mke2fs and mkfs.ext2

mke2fs is used to create a Linux second extended file system on a device (usually a disk partition). mkfs.ext2 does the same as mke2fs.

mklost+found

mklost+found is used to create a lost+found directory in the current working directory on a Linux second extended file system. mklost+found pre-allocates disk blocks to the directory to make it usable by e2fsck.

tune2fs

tune2fs adjusts tunable filesystem parameters on a Linux second extended filesystem.

File

Contents

The File package contains the file program.

Description

File tests each specified file in an attempt to classify it. There are three sets of tests, performed in this order: filesystem tests, magic number tests, and language tests. The first test that succeeds causes the file type to be printed.

Fileutils

Contents

The Fileutils package contains the chgrp, chmod, chown, cp, dd, df, dir, dircolors, du, install, ln, ls, mkdir, mkfifo, mknod, mv, rm, rmdir, sync, touch and vdir programs.

Description

chgrp

chgrp changes the group ownership of each given file to the named group, which can be either a group name or a numeric group ID.

chmod

chmod changes the permissions of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new permissions.

chown

chown changes the user and/or group ownership of each given file.

cp

cp copies files from one place to another.

dd

dd copies a file (from the standard input to the standard output, by default) with a user-selectable blocksize, while optionally performing conversions on it.

df

df displays the amount of disk space available on the filesystem containing each file name argument. If no file name is given, the space available on all currently mounted filesystems is shown.

ls, dir and vdir

dir and vdir are versions of ls with different default output formats. These programs list each given file or directory name. Directory contents are sorted alphabetically. For ls, files are by default listed in columns, sorted vertically, if the standard output is a terminal; otherwise they are listed one per line. For dir, files are by default listed in columns, sorted vertically. For vdir, files are by default listed in long format.

dircolors

dircolors outputs commands to set the LS_COLOR environment variable. The LS_COLOR variable is used to change the default color scheme used by ls and related utilities.

du

du displays the amount of disk space used by each argument and for each subdirectory of directory arguments.

install

install copies files and sets their permission modes and, if possible, their owner and group.

ln

ln makes hard or soft (symbolic) links between files.

mkdir

mkdir creates directories with a given name.

mkfifo

mkfifo creates a FIFO with each given name.

mknod

mknod creates a FIFO, character special file, or block special file with the given file name.

mv

mv moves files from one directory to another or renames files, depending on the arguments given to mv.

rm

rm removes files or directories.

rmdir

rmdir removes directories, if they are empty.

sync

sync forces changed blocks to disk and updates the super block.

touch

touch changes the access and modification times of each given file to the current time. Files that do not exist are created empty.

Gettext

Contents

The gettext package contains the `gettext`, `gettextize`, `msgcmp`, `msgcomm`, `msgfmt`, `msgmerge`, `msgunfmt` and `xgettext` programs.

Description

gettext

The gettext package is used for internationalization (also known as i18n) and for localization (also known as l10n). Programs can be compiled with Native Language Support (NLS) which enable them to output messages in your native language rather than in the default English language.

Grep

Contents

The grep package contains the egrep, fgrep and grep programs.

Description

egrep

egrep prints lines from files matching an extended regular expression pattern.

fgrep

fgrep prints lines from files matching a list of fixed strings, separated by newlines, any of which is to be matched.

grep

grep prints lines from files matching a basic regular expression pattern.

Gzip

Contents

The Gzip package contains the gunzip, gzexe, gzip, zcat, zcmp, zdiff, zforce, zgrep, zmore and znew programs.

Description

gunzip

gunzip decompresses files that are compressed with gzip.

gzexe

gzexe allows you to compress executables in place and have them automatically uncompress and execute when you run them (at a penalty in performance).

gzip

gzip reduces the size of the named files using Lempel–Ziv coding (LZ77).

zcat

zcat uncompresses either a list of files on the command line or its standard input and writes the uncompressed data on standard output

zcmp

zcmp invokes the cmp program on compressed files.

zdiff

zdiff invokes the diff program on compressed files.

zforce

zforce forces a .gz extension on all gzip files so that gzip will not compress them twice. This can be useful for files with names truncated after a file transfer.

zgrep

zgrep invokes the grep program on compressed files.

zmore

Zmore is a filter which allows examination of compressed or plain text files one screenful at a time on a soft-copy terminal (similar to the more program).

znew

Znew recompresses files from .Z (compress) format to .gz (gzip) format.

Ld.so

Contents

From the Ld.so package we're using the ldconfig and ldd programs.

Description

ldconfig

ldconfig creates the necessary links and cache (for use by the run-time linker, ld.so) to the most recent shared libraries found in the directories specified on the command line, in the file /etc/ld.so.conf, and in the trusted directories (/usr/lib and /lib). ldconfig checks the header and file names of the libraries it encounters when determining which versions should have their links updated.

ldd

ldd prints the shared libraries required by each program or shared library specified on the command line.

Libtool

Contents

The Libtool package contains the libtool and libtoolize programs. It also contains the ltdl library.

Description

libtool

Libtool provides generalized library-building support services.

libtoolize

libtoolize provides a standard way to add libtool support to your package.

ltdl library

Libtool provides a small library, called 'libltdl', that aims at hiding the various difficulties of dloping libraries from programmers.

Bin86

Contents

The Bin86 contains the as86, as86_encap, ld86, objdump86, nm86 and size86 programs.

Description

as86

as86 is an assembler for the 8086...80386 processors.

as86_encap

as86_encap is a shell script to call as86 and convert the created binary into a C file prog.v to be included in or linked with programs like boot block installers.

ld86

ld86 understands only the object files produced by the as86 assembler, it can link them into either an impure or a separate I&D executable.

objdump86

No description available.

nm86

No description available.

size86

No description available.

Lilo

Contents

The Lilo package contains the lilo program.

Description

lilo installs the Linux boot loader which is used to start a Linux system.

Make

Contents

The Make package contains the make program.

Description

make determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them.

Shellutils

Contents

The Shellutils package contains the `basename`, `chroot`, `date`, `dirname`, `echo`, `env`, `expr`, `factor`, `false`, `groups`, `hostid`, `hostname`, `id`, `logname`, `nice`, `nohup`, `pathchk`, `pinky`, `printenv`, `printf`, `pwd`, `seq`, `sleep`, `stty`, `su`, `tee`, `test`, `true`, `tty`, `uname`, `uptime`, `users`, `who`, `whoami` and `yes` programs.

Description

basename

`basename` strips directory and suffixes from filenames.

chroot

`chroot` runs a command or interactive shell with special root directory.

date

`date` displays the current time in a specified format, or sets the system `date`.

dirname

`dirname` strips non–directory suffixes from file name.

echo

`echo` displays a line of text.

env

`env` runs a program in a modified environment.

expr

`expr` evaluates expressions.

factor

factor prints the prime factors of all specified integer numbers.

false

false always exits with a status code indicating failure.

groups

groups prints the groups a user is in.

hostid

hostid prints the numeric identifier (in hexadecimal) for the current host.

hostname

hostname sets or prints the name of the current host system

id

id prints the real and effective UIDs and GIDs of a user or the current user.

logname

logname prints the current user's login name.

nice

nice runs a program with modified scheduling priority.

nohup

nohup runs a command immune to hangups, with output to a non-tty

pathchk

pathchk checks whether file names are valid or portable.

pinky

pinky is a lightweight finger utility which retrieves information about a certain user

printenv

printenv prints all or part of the environment.

printf

printf formats and print data (the same as the printf C function).

pwd

pwd prints the name of the current/working directory

seq

seq prints numbers in a certain range with a certain increment.

sleep

sleep delays for a specified amount of time.

stty

stty changes and prints terminal line settings.

su

su runs a shell with substitute user and group IDs

tee

tee reads from standard input and write to standard output and files.

test

test checks file types and compares values.

true

True always exitx with a status code indicating success.

tty

tty prints the file name of the terminal connected to standard input.

uname

uname prints system information.

uptime

uptime tells how long the system has been running.

users

users prints the user names of users currently logged in to the current host.

who

who shows who is logged on.

whoami

whoami prints your effective userid.

yes

yes outputs a string repeatedly until killed.

Shadow Password Suite

Contents

The Shadow Password Suite contains the `chage`, `chfn`, `chsh`, `expiry`, `faillog`, `gpasswd`, `lastlog`, `login`, `newgrp`, `passwd`, `sg`, `su`, `chpasswd`, `dpasswd`, `groupadd`, `groupdel`, `groupmod`, `grpck`, `grpconv`, `grpunconv`, `logoutd`, `mkpasswd`, `newusers`, `pwck`, `pwconv`, `pwunconv`, `useradd`, `userdel`, `usermod` and `vipw` programs.

Description

chage

`chage` changes the number of days between password changes and the date of the last password change.

chfn

`chfn` changes user fullname, office number, office extension, and home phone number information for a user's account.

chsh

`chsh` changes the user login shell.

expiry

It's currently unknown what this program is for.

faillog

`faillog` formats the contents of the failure log, `/var/log/faillog`, and maintains failure counts and limits.

gpasswd

`gpasswd` is used to administer the `/etc/group` file

lastlog

`lastlog` formats and prints the contents of the last login log, `/var/log/lastlog`. The login-name, port, and last login time will be printed.

login

login is used to establish a new session with the system.

newgrp

newgrp is used to change the current group ID during a login session.

passwd

passwd changes passwords for user and group accounts.

sg

sg executes command as a different group ID.

su

Change the effective user id and group id to that of a user. This replaces the su programs that's installed from the Shellutils package.

chpasswd

chpasswd reads a file of user name and password pairs from standard input and uses this information to update a group of existing users.

dpasswd

dpasswd adds, deletes, and updates dialup passwords for user login shells.

groupadd

The groupadd command creates a new group account using the values specified on the command line and the default values from the system.

groupdel

The `groupdel` command modifies the system account files, deleting all entries that refer to group.

groupmod

The `groupmod` command modifies the system account files to reflect the changes that are specified on the command line.

grpck

`grpck` verifies the integrity of the system authentication information.

grpconv

`grpconv` converts to shadow group files from normal group files.

grpunconv

`grpunconv` converts from shadow group files to normal group files.

logoutd

`logoutd` enforces the login time and port restrictions specified in `/etc/porttime`.

mkpasswd

`mkpasswd` reads a file in the format given by the flags and converts it to the corresponding database file format.

newusers

`newusers` reads a file of user name and cleartext password pairs and uses this information to update a group of existing users or to create new users.

pwck

`pwck` verifies the integrity of the system authentication information.

pwconv

pwconv converts to shadow passwd files from normal passwd files.

pwunconv

pwunconv converts from shadow passwd files to normal files.

useradd

useradd creates a new user or update default new user information.

userdel

userdel modifies the system account files, deleting all entries that refer to a specified login name.

usermod

usermod modifies the system account files to reflect the changes that are specified on the command line.

vipw and vigr

vipw and vigr will edit the files /etc/passwd and /etc/group, respectively. With the `-s` flag, they will edit the shadow versions of those files, /etc/shadow and /etc/gshadow, respectively.

Modutils

Contents

The Modutils package contains the depmod, genksyms, insmod, insmod_ksymoops_clean, kerneld, kernelversion, ksyms, lsmod, modinfo, modprobe and rmmod programs.

Description

depmod

depmod handles dependency descriptions for loadable kernel modules.

genksyms

genksyms reads (on standard input) the output from `gcc -E source.c` and generates a file containing version information.

insmod

insmod installs a loadable module in the running kernel.

insmod_ksymoops_clean

insmod_ksymoops_clean deletes saved ksyms and modules not accessed in 2 days.

kerneld

kerneld performs kernel action in user space (such as on-demand loading of modules)

kernelversion

kernelversion reports the major version of the running kernel.

ksyms

ksyms displays exported kernel symbols.

lsmod

lsmod shows information about all loaded modules.

modinfo

modinfo examines an object file associated with a kernel module and displays any information that it can glean.

modprobe

Modprobe uses a Makefile-like dependency file, created by depmod, to automatically load the relevant module(s) from the set of modules available in predefined directory trees.

rmmod

rmmod unloads loadable modules from the running kernel.

Procinfo

Contents

The Procinfo package contains the procinfo program.

Description

procinfo gathers some system data from the /proc directory and prints it nicely formatted on the standard output device.

Procps

Contents

The Procps package contains the free, kill, oldps, ps, skill, snice, sysctl, tload, top, uptime, vmstat, w and watch programs.

Description

free

free displays the total amount of free and used physical and swap memory in the system, as well as the shared memory and buffers used by the kernel.

kill

kill sends signals to processes.

oldps and ps

ps gives a snapshot of the current processes.

skill

skill sends signals to process matching a criteria.

snice

snice changes the scheduling priority for process matching a criteria.

sysctl

sysctl modifies kernel parameters at runtime.

tload

tload prints a graph of the current system load average to the specified tty (or the tty of the tload process if none is specified).

top

top provides an ongoing look at processor activity in real time.

uptime

uptime gives a one line display of the following information: the current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

vmstat

vmstat reports information about processes, memory, paging, block IO, traps, and cpu activity.

w

w displays information about the users currently on the machine, and their processes.

watch

watch runs command repeatedly, displaying its output (the first screenfull).

Vim

Contents

The Vim package contains the ctags, etags, ex, gview, gvim, rgview, rgvim, rview, rvim, view, vim, vintutor and xxd programs.

Description

ctags

ctags generate tag files for source code.

etags

etags does the same as ctags but it can generate cross reference files which list information about the various source objects found in a set of lanugage files.

ex

ex starts vim in Ex mode.

gview

gview is the GUI version of view.

gvim

gvim is the GUI version of vim.

rgview

rgview is teh GUI version of rview.

rgvim

rgvim is the GUI version of rvim.

rview

rview is a restricted version of view. No shell commands can be started and Vim can't be suspended.

rvim

rvim is the restricted version of vim. No shell commands can be started and Vim can't be suspended.

view

view starts vim in read-only mode.

vim

vim starts vim in the normal, default way.

vimtutor

vimtutor starts the Vim tutor.

xxd

xxd makes a hexdump or does the reverse.

Psmisc

Contents

The Psmisc package contains the fuser, killall and pstree programs.

Description

fuser

fuser displays the PIDs of processes using the specified files or file systems.

killall

killall sends a signal to all processes running any of the specified commands.

pstree

pstree shows running processes as a tree.

Sed

Contents

The Sed package contains the sed program.

Description

sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline).

Sysklogd

Contents

The Sysklogd package contains the klogd and syslogd programs.

Description

klogd

klogd is a system daemon which intercepts and logs Linux kernel messages.

syslogd

Syslogd provides a kind of logging that many modern programs use. Every logged message contains at least a time and a hostname field, normally a program name field, too, but that depends on how trustworthy the logging program is.

Sysvinit

Contents

The Sysvinit package contains the pidof, last, lastb, mesg, utmpdump, wall, halt, init, killall5, poweroff, reboot, runlevel, shutdown, sulogin and telinit programs.

Description

pidof

Pidof finds the process id's (pids) of the named programs and prints those id's on standard output.

last

last searches back through the file /var/log/wtmp (or the file designated by the -f flag) and displays a list of all users logged in (and out) since that file was created.

lastb

lastb is the same as last, except that by default it shows a log of the file /var/log/btmp, which contains all the bad login attempts.

mesg

Mesg controls the access to your terminal by others. It's typically used to allow or disallow other users to write to your terminal.

utmpdump

utmpdumps prints the content of a file (usually /var/run/utmp) on standard output in a user friendly format.

wall

Wall sends a message to everybody logged in with their mesg permission set to yes.

halt

Halt notes that the system is being brought down in the file `/var/log/wtmp`, and then either tells the kernel to halt, reboot or `poweroff` the system. If `halt` or `reboot` is called when the system is not in runlevel 0 or 6, `shutdown` will be invoked instead (with the flag `-h` or `-r`).

init

Init is the parent of all processes. Its primary role is to create processes from a script stored in the file `/etc/inittab`. This file usually has entries which cause `init` to spawn `getty`s on each line that users can log in. It also controls autonomous processes required by any particular system.

killall5

`killall5` is the SystemV `killall` command. It sends a signal to all processes except the processes in its own session, so it won't kill the shell that is running the script it was called from.

poweroff

`poweroff` is equivalent to `shutdown -h -p now`. It halts the computer and switches off the computer (when using an APM compliant BIOS and APM is enabled in the kernel).

reboot

`reboot` is equivalent to `shutdown -r now`. It reboots the computer.

runlevel

`Runlevel` reads the system `utmp` file (typically `/var/run/utmp`) to locate the runlevel record, and then prints the previous and current system runlevel on its standard output, separated by a single space.

shutdown

`shutdown` brings the system down in a secure way. All logged-in users are notified that the system is going down, and login is blocked.

sulogin

`sulogin` is invoked by `init` when the system goes into single user mode (this is done through an entry in `/etc/inittab`). `Init` also tries to execute `sulogin` when it is passed the `-b` flag from the bootmonitor (eg, LILO).

telinit

telinit sends appropriate signals to init, telling it which runlevel to change to.

Tar

Contents

The tar package contains the tar and rmt programs.

Description

tar

tar is an archiving program designed to store and extract files from an archive file known as a tarfile.

rmt

rmt is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection.

Textutils

Contents

The Textutils package contains the cat, cksum, comm, split, cut, expand, fmt, fold, head, join, md5sum, nl, od, paste, pr, ptx, sort, split, sum, tac, tail, tr, tsort, unexpand, uniq and wc programs.

Description

cat

cat concatenates file(s) or standard input to standard output.

cksum

cksum prints CRC checksum and byte counts of each specified file.

comm

comm compares two sorted files line by line.

csplit

csplit outputs pieces of a file separated by (a) pattern(s) to files xx01, xx02, ..., and outputs byte counts of each piece to standard output.

cut

cut prints selected parts of lines from specified files to standard output.

expand

expand converts tabs in files to spaces, writing to standard output.

fmt

fmt reformats each paragraph in the specified file(s), writing to standard output.

fold

fold wraps input lines in each specified file (standard input by default), writing to standard output.

head

Print first xx (10 by default) lines of each specified file to standard output.

join

join joins lines of two files on a common field.

md5sum

md5sum prints or checks MD5 checksums.

nl

nl writes each specified file to standard output, with line numbers added.

od

od writes an unambiguous representation, octal bytes by default, of a specified file to standard output.

paste

paste writes lines consisting of the sequentially corresponding lines from each specified file, separated by TABs, to standard output.

pr

pr paginates or columnates files for printing.

ptx

ptx produces a permuted index of file contents.

sort

sort writes sorted concatenation of files to standard output.

split

split outputs fixed-size pieces of an input file to PREFIXaa, PREFIXab, ...

sum

sum prints checksum and block counts for each specified file.

tac

tac writes each specified file to standard output, last line first.

tail

tail print the last xx (10 by default) lines of each specified file to standard output.

tr

tr translates, squeezes, and/or deletes characters from standard input, writing to standard output.

tsort

tsort writes totally ordered lists consistent with the partial ordering in specified files.

unexpand

unexpand converts spaces in each file to tabs, writing to standard output.

uniq

uniq discards all but one of successive identical lines from files or standard input and writes to files or standard output.

WC

wc prints line, word, and byte counts for each specified file, and a total line if more than one file is specified.

Util Linux

Contents

The Util-linux package contains the arch, dmesg, kill, more, mount, umount, agetty, blockdev, cfdisk, ctrlaltdel, elvtune, fdisk, fsck.minix, hwclock, kbdrate, losetup, mkfs, mkfs.bfs, mkfs.minix, mkswap, sfdisk, swapoff, swapon, cal, chkdupexe, col, colcrt, colrm, column, cytune, ddate, fdformat, getopt, hexdump, ipcrm, ipcs, logger, look, mcookie, namei, rename, renice, rev, script, setfdprm, setid, setterm, ul, whereis, write, ramsize, rdev, readprofile, rootflags, swapdev, tunelp and vidmode programs.

Description

arch

arch prints the machine architecture.

dmesg

dmesg is used to examine or control the kernel ring buffer (boot messages from the kernel).

kill

kill sends a specified signal to the specified process.

more

more is a filter for paging through text one screenful at a time.

mount

mount mounts a filesystem from a device to a directory (mount point).

umount

umount unmounts a mounted filesystem.

agetty

agetty opens a tty port, prompts for a login name and invokes the /bin/login command.

blockdev

No description available.

fdisk

fdisk is an libncurses based disk partition table manipulator.

ctrlaltdel

ctrlaltdel sets the function of the CTRL+ALT+DEL key combination (hard or soft reset).

elvtune

elvtune allows to tune the I/O elevator per blockdevice queue basis.

fdisk

fdisk is a disk partition table manipulator.

fsck.minix

fsck.minix performs a consistency check for the Linux MINIX filesystem.

hwclock

hwclock queries and sets the hardware clock (Also called the RTC or BIOS clock).

kbdrate

kbdrate resets the keyboard repeat rate and delay time.

losetup

losetup sets up and controls loop devices.

mkfs

mkfs builds a Linux filesystem on a device, usually a harddisk partition.

mkfs.bfs

mkfs.bfs creates a SCO bfs file system on a device, usually a harddisk partition.

mkfs.minix

mkfs.minix creates a Linux MINIX filesystem on a device, usually a harddisk partition.

mkswap

mkswap sets up a Linux swap area on a device or in a file.

sfdisk

sfdisk is a disk partition table manipulator.

swapoff

swapoff disables devices and files for paging and swapping.

swapon

swapon enables devices and files for paging and swapping.

cal

cal displays a simple calender.

chkdupexe

chkdupexe finds duplicate executables.

col

col filters reverse line feeds from input.

colcrt

colcrt filters nroff output for CRT previewing.

colrm

colrm removes columns from a file.

column

column columnates lists.

cytune

cytune queries and modifies the interruption threshold for the Cyclades driver.

ddate

ddate converts Gregorian dates to Discordian dates.

fdformat

fdformat low-level formats a floppy disk.

getopt

getops parses command options the same way as the getopt C command.

hexdump

hexdump displays specified files, or standard input, in a user specified format (ascii, decimal, hexadecimal, octal).

ipcrm

ipcrm removes a specified resource.

ipcs

ipcs provides information on ipc facilities.

logger

logger makes entries in the system log.

look

look displays lines beginning with a given string.

mcookie

mcookie generates magic cookies for xauth.

namei

namei follows a pathname until a terminal point is found.

rename

rename renames files.

renice

renice alters priority of running processes.

rev

rev reverses lines of a file.

script

script makes typescript of terminal session.

setfdprm

setfdprm sets user-provides floppy disk parameters.

setsid

setsid runs programs in a new session.

setterm

setterm sets terminal attributes.

ul

ul reads a file and translates occurrences of underscores to the sequence which indicates underlining for the terminal in use.

whereis

whereis locates a binary, source and manual page for a command.

write

write sends a message to another user.

ramsize

ramsize queries and sets RAM disk size.

rdev

rdev queries and sets image root device, swap device, RAM disk size, or video mode.

readprofile

readprofile reads kernel profiling information.

rootflags

rootflags queries and sets extra information used when mounting root.

swapdev

swapdev queries and sets swap device.

tunelp

tunelp sets various parameters for the lp device.

vidmode

vidmode queries and sets the video mode.

Console-tools

Contents

The Console-tools package contains the charset, chvt, consolechars, deallocvt, dumpkeys, fgconsole, fix_bs_and_del, font2psf, getkeycodes, kbd_mode, loadkeys, loadunimap, mapscrn, mk_modmap, openvt, psfaddtable, psfgettable, psfstrietable, resizecons, saveunimap, screendump, setfont, setkeycodes, setleds, setmetamode, setvesablank, showcfont, showkey, splitfont, unicode_start, unicode_stop, vcstime, vt-is-URF8, writevt

Description

charset

charset sets an ACM for use in one of the G0/G1 charsets slots.

chvt

chvt changes foreground virtual terminal.

codepage

No description available.

consolechars

consolechars loads EGA/VGA console screen fonts, screen font maps and/or application-charset maps.

deallocvt

deallocvt deallocates unused virtual terminals.

dumpkeys

dumpkeys dumps keyboard translation tables.

fgconsole

fgconsole prints the number of the active virtual terminal.

fix_bs_and_del

No description available.

font2psf

No description available.

getkeycodes

getkeycodes prints the kernel scancode-to-keycode mapping table.

kbd_mode

kbd_mode reports or sets the keyboard mode.

loadkeys

loadkeys loads keyboard translation tables.

loadunimap

No description available.

mapscrn

No description available.

mk_modmap

No description available.

openvt

openvt starts a program on a new virtual terminal.

psfaddtable

psfaddtable adds a Unicode character table to a console font.

psfgettable

psfgettable extracts the embedded Unicode character table from a console font.

psfstriptable

psfstriptable removes the embedded Unicode character table from a console font.

resizecons

resizecons changes the kernel idea of the console size.

saveunimap

No description available.

screendump

No description available.

setfont

No description available.

setkeycodes

setkeycodes loads kernel scancode-to-keycode mapping table entries.

setleds

setleds sets the keyboard leds.

setmetamode

setmetamode defines the keyboard meta key handling.

setvesablank

No description available.

showcfont

showcfont displays all character in the current screenfont.

showkey

showkey examines the scancodes and keycodes sent by the keyboard.

splitfont

No description available.

unicode_start

unicode_start puts the console in Unicode mode.

unicode_stop

No description available.

vcstime

No description available.

vt-is-UTF8

vt-is-UTF8 checks whether the current virtual terminal is in UTF8- or byte-mode.

writetv

No description available.

Console-data

Contents

The console-data package contains the data files that are used and needed by the console-tools package.

Man–pages

Contents

The Man–pages package contains various manual pages that don't come with the packages.

Description

Examples of provided manual pages are the manual pages describing all the C and C++ functions, few important /dev/ files and more.

Appendix B. Resources

Introduction

A list of books, HOWTOs and other documents you might find useful to download or buy follows. This list is just a small list to start with. We hope to be able to expand this list in time as we come across more useful documents or books.

Books

- Linux Network Administrator's Guide published by O'Reilly. ISBN: 1-56502-087-2
 - Running Linux published by O'Reilly. ISBN: 1-56592-151-8
-

HOWTOs and Guides

All of the following HOWTOs can be downloaded from the Linux Documentation Project site at <http://www.linuxdoc.org>

- Linux Network Administrator's Guide
 - Powerup2Bash-HOWTO
-

Other

- The various man and info pages that come with the packages
-

Appendix C. Official download locations

Below you find the list with packages from chapter 3 with their original download locations. This might help you to find a newer version of a package quicker.

- Bash (2.04): <ftp://ftp.gnu.org/gnu/bash/>
- Binutils (2.10): <ftp://ftp.gnu.org/gnu/binutils/>
- Bzip2 (1.0.1): <ftp://sourceware.cygnum.com/pub/bzip2/>
- Diff Utils (2.7): <ftp://ftp.gnu.org/gnu/diffutils/>
- File Utils (4.0): <ftp://ftp.gnu.org/gnu/fileutils/>
- GCC (2.95.2): <ftp://ftp.gnu.org/gnu/gcc/>
- Linux Kernel (2.2.17): <ftp://ftp.kernel.org/pub/linux/kernel/>
- Glibc (2.1.3): <ftp://ftp.gnu.org/gnu/glibc/>
- Glibc-crypt (2.1.3): <ftp://ftp.gnu.org/gnu/glibc/>
- Glibc-linuxthreads (2.1.3): <ftp://ftp.gnu.org/gnu/glibc/>
- Glibc Patch (2.1.3): <ftp://packages.linuxfromscratch.org/common-packages>
- Grep (2.4.2): <ftp://ftp.gnu.org/gnu/grep/>
- Gzip (1.2.4a): <ftp://ftp.gnu.org/gnu/gzip/>
- Make (3.79.1): <ftp://ftp.gnu.org/gnu/make/>
- Sed (3.02): <ftp://ftp.gnu.org/gnu/sed/>
-

- Sh–utils (2.0): <ftp://ftp.gnu.org/gnu/sh-utils/>
-
- Tar (1.13): <ftp://ftp.gnu.org/gnu/tar/>
-
- Text Utils (2.0): <ftp://ftp.gnu.org/gnu/textutils/>
-
- MAKEDEV (2.5): <ftp://ftp.ihg.uni-duisburg.de/Linux/system/>
-
- Bison (1.28): <ftp://ftp.gnu.org/gnu/bison/>
-
- Mawk (1.3.3): <ftp://ftp.whidbey.net/pub/brennan/>
-
- Patch (2.5.4): <ftp://ftp.gnu.org/gnu/patch/>
-
- Find Utils (4.1): <ftp://ftp.gnu.org/gnu/findutils/>
-
- Find Utils Patch (4.1): ftp://packages.linuxfromscratch.org/pub/common-packages
-
- Ncurses (5.1): <ftp://ftp.gnu.org/gnu/ncurses/>
-
- Less (358): <ftp://ftp.gnu.org/gnu/less/>
-
- Groff (1.16.1): <ftp://ftp.gnu.org/gnu/groff/>
-
- Man (1.5h1): <ftp://ftp.win.tue.nl/pub/linux-local/utils/man/>
-
- Perl (5.6.0): <http://www.perl.com>
-
- M4 (1.4): <ftp://ftp.gnu.org/gnu/m4/>
-
- Texinfo (4.0): <ftp://ftp.gnu.org/gnu/texinfo/>
-
- Autoconf (2.13): <ftp://ftp.gnu.org/gnu/autoconf/>
-

- Automake (1.4): <ftp://ftp.gnu.org/gnu/automake/>
-
- Flex (2.5.4a): <ftp://ftp.gnu.org/non-gnu/flex/>
-
- File (3.32): <ftp://ftp.gw.com/mirrors/pub/unix/file/>
-
- Libtool (1.3.5): <ftp://ftp.gnu.org/gnu/libtool/>
-
- Bin86 (0.15.1): <http://www.cix.co.uk/~mayday/>
-
- Gettext (0.10.35): <ftp://ftp.gnu.org/gnu/gettext/>
-
- Console-tools (0.2.3): <ftp://metalab.unc.edu/pub/Linux/system/keyboards/>
-
- Console-tools Patch (0.2.3): <ftp://packages.linuxfromscratch.org/pub/common-packages>
-
- Console-data (1999.08.29): <ftp://metalab.unc.edu/pub/Linux/system/keyboards/>
-
- E2fsprogs (1.19): <ftp://download.sourceforge.net/pub/sourceforge/e2fsprogs/>
-
- Ed (0.2): <ftp://ftp.gnu.org/gnu/ed/>
-
- Ld.so (1.9.9): <ftp://packages.linuxfromscratch.org/pub/common-packages>
-
- Lilo (21.6): <ftp://brun.dyndns.org/pub/linux/lilo>
-
- Modutils (2.3.17): <ftp://ftp.kernel.org/pub/linux/utils/kernel/modutils>
-
- Vim-rt (5.7) 1,073 KB: <ftp://ftp.vim.org/pub/editors/vim/unix/>
-
- Vim-src (5.7) 1,202 KB: <ftp://ftp.vim.org/pub/editors/vim/unix/>
-
- Procinfo (17): <ftp://ftp.cistron.nl/pub/people/svm/>
-

Procps (2.0.7): <ftp://people.redhat.com/johnsonm/procps/>

-
- Psmisc (19): <ftp://lrcftp.epfl.ch/pub/linux/local/psmisc/>
-
- Shadow Password Suite (20000902): <ftp://ftp.ists.pwr.wroc.pl/pub/linux/shadow/>
-
- Sysklogd (1.4): <ftp://sunsite.unc.edu/pub/Linux/system/daemons/>
-
- Sysvinit (2.78): <ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/>
-
- Util Linux (2.10o): <ftp://ftp.win.tue.nl/pub/linux/utils/util-linux/>
-
- Man-pages (1.31): <ftp://ftp.win.tue.nl/pub/linux/docs/manpages/>
-
- Netkit-base (0.17): <ftp://ftp.uk.linux.org/pub/linux/Networking/netkit/>
-
- Net-tools (1.57): <http://www.tazenda.demon.co.uk/phil/net-tools/>