Linux From Scratch

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Linux From Scratch

Gerard Beekmans − Main document

Michael Peters − Apple PowerPC additions

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.

Dedication

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

Table of Contents

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 nothing but the sources of software that are needed.

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 yourself and install them. That could be another reason why you would want to build a custom made Linux system.

For those and numerous other reasons somebody might want to build his or her own Linux system from the ground up. If you are one of those people, this book is meant for you.

Who would not want to read this book

<here what Greg O'Keefe is going to write up>

Organization

This book is devided 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 a basic system on Intel systems

Part Two guides you through the installation of a basic system on Intel systems 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 – Installation of a basic system on Apple PowerPC systems

Part Three is the Apple PowerPC version of part two.

Part IV – Appendixes

Part Four contains various Appendixes.

I. Part I – Introduction

Table of Contents

Chapter 1. Introduction

What's this all about?

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.

Book versions

This is Development version 2.3.2 dated May 4th, 2000. If this version is older than a month you definitely want to take a look at our website and download a newer version. Development versions are released once every two to three weeks. Stable versions are released once every one to two months.

The latest versions of this book and related files can be downloaded from one of the following sites. Please avoid the main site at Dallas whenever possible. Thanks.

- Dallas, Texas, United States – http://www.linuxfromscratch.org
- Columbus, Ohio, United States – http://lfs.bcpub.com
- United States – http://clueserver.org/lfs
- Braunschweig, Niedersachsen, Germany – http://134.169.139.209
- Brisbane, Queensland, Australia – http://lfs.mirror.aarnet.edu.au

Credits

First of all I would like to thank the following persons for hosting and mirroring the linuxfromscratch.org services:

- Paul Jensen for providing http://www.pcrdallas.com as the main linuxfromscratch.org host
- Bryan Dumm for providing http://www.bcpub.com as the lfs.bcpub.com mirror
- Alan Olsen for providing http://clueserver.org as the clueserver.org/lfs mirror
- Jan Niemann for providing http://helga.lk.etc.tu−bs.de as the 134.169.139.209 mirror
- Jason Andrade for providing http://mirror.aarnet.edu.au as the lfs.mirror.aarnet.edu.au mirror

Changelog

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 occured during the transistion 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 it's own dedicated chapters now. This should increase readability a lot

- All optional chapters have been removed for now. These chapters are going to be restructured into dedicated parts such as a chapter that deals with setting up LFS as an email server. A chapter that deals with setting up LFS as a http server, and so forth. These reorganizations couldn't make this development version in time. So you'll have to read the current stable 2.2 version of this book for those parts.

- 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 three public accessible mailinglists:

- lfs−discuss
- lfs−announce
- linux

lfs−discuss

The lfs−discuss list is the list that discusses matters regarding this book. If you have problems, comments, suggestions, etc. join this list and post your message. People on this list can take part in the newest developments regarding this book.

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.

linux

The linux list is a general Linux discussion list that handles everything that has got anything to do with Linux in any way, shape and form. This list was created originally to stop the high volume of off−topic messages to the lfs−discuss list. Although some of the messages posted to the linux are somewhat related to this book, the list is also used for anything else that isn't related to this book at all. Feel free to join this list if you have non−LFS questions or just want to discuss a subject.

How to subscribe?

You can subscribe to any of the above mentioned mailinglists by sending an email to majordomo@linuxfromscratch.org and write *subscribe listname* in the body of the message, where listname is replaced by either lfs−discuss, lfs−announce or linux. No subject required.

You can, if you want, subscribe to multiple lists at the same time using one email. Just repeat the subscribe command for each of the lists you want to subscribe to.

After you have sent the email, the Majordomo 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 particulair list.

How to unsubscribe?

To unsubscribe from a list, send an email to majordomo@linuxfromscratch.org and write *unsubscribe listname* in the body of the message, where listname is replaced by either lfs−discuss, lfs−announce or linux.

You can, if you want, unsubscribe from multiple lists at the same time using one email. Just repeat the ubsubscribe command for each of the lists you want to unsubscribe from.

Mail archives

There is a mailinglist archive for the lfs−discuss and linux mailinglists. The lfs−discuss mailinglist archive can be found at http://www.pcrdallas.com/mail−archives/lfs−discuss and the linux mailinglist archive can be found at http://www.pcrdallas.com/mail−archives/linux

---

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 personaly, send an email to mpters@mac.com

Chapter 2. Important information

About $LFS

Please read the following carefully: throughout this document 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 later on in full detail in chapter 4. In my case the LFS partition is mounted on /mnt/hda5. If I read this document myself and I see $LFS somewhere, I will pretend that I read /mnt/hda5. If I read that I have to run this command: cp inittab $LFS/etc I actually will run this: cp inittab /mnt/hda5/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 some 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/hda5. This is accomplished by running: export LFS=/mnt/hda5

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/hda5/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 break your system badly and might have to reinstall it if you don't know how to repair it. So that's why I strongly advise against using the $LFS variable. You better replace $LFS yourself by something like /mnt/hda5. If you make a typo while entering /mnt/hda5, the worst thing that can happen is that you'll get an error saying "no such file or directory" but it won't break your system. Don't say I didn't warn you ;)

How to download the software

Throughout this document I will assume that you have stored all the packages you have downloaded in a subdirectory under $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−3.02.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/<somesubdirectory> when you have finished the chapter in which you prepare a new partition (which chapter exactly depends on your architecture).

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 see this from a .tar.gz or .tgz extension). I'm not going to write down every time how to ungzip 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:

```
root:~# 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:

```
root:/usr/src# tar xvfz filename.tar.gz
root:/usr/src# tar xvfz filename.tgz
```

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

```
root:/usr/src#    tar --use-compress-prog=bzip2 -xvf
filename.tar.bz2
```

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

```
root:/usr/src# 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. I'm not going to tell you every time to unpack it.

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 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.

II. Part II – Installation of a basic system on Intel systems

Table of Contents

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 download a newer version).

Please note that this list used to be ordered on usage, meaning that the first package mentioned in this list was the first package used in this book. That's no longer the case because several chapters have been moved around, so that doens't apply. I didn't have the time to re−order this list in this development release. The next release will have this list ordered again.

- Sysvinit (2.78): ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/
- Bash (2.04): ftp://ftp.gnu.org/gnu/bash
- Linux Kernel (2.2.14): ftp://ftp.kernel.org/pub/linux/kernel/
- Binutils (2.9.5.0.37): ftp://ftp.varesearch.com/pub/support/hjl/binutils/
- Bzip2 (0.9.5d): http://sourceware.cygnus.com/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/
- Glibc (2.1.3): ftp://ftp.gnu.org/gnu/glibc/
- Glibc−crypt (2.1.3): ftp://ftp.gwdg.de/pub/linux/glibc/
- Glibc−linuxthreads (2.1.3): ftp://ftp.gnu.org/gnu/glibc/
- Grep (2.4.2): ftp://ftp.gnu.org/gnu/grep/
- Gzip (1.2.4a): ftp://ftp.gnu.org/gnu/gzip/

Make (3.78.1): ftp://ftp.gnu.org/gnu/make/

- Ed (0.2): ftp://ftp.gnu.org/gnu/ed/

- Patch (2.5.4): ftp://ftp.gnu.org/gnu/patch/

- Sed (3.02): ftp://ftp.gnu.org/gnu/sed/

- Shell 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/

- Util Linux (2.10h): ftp://ftp.win.tue.nl/pub/linux/utils/util−linux/

- Bison (1.28): ftp://ftp.gnu.org/gnu/bison/

- Mawk (1.3.3) ftp://ftp.whidbey.net/pub/brennan/

- Find Utils (4.1): ftp://ftp.gnu.org/gnu/findutils/

- Termcap (1.3): ftp://ftp.gnu.org/gnu/termcap/

- Ncurses (5.0): ftp://ftp.gnu.org/gnu/ncurses/

- Less (340): ftp://ftp.gnu.org/gnu/less/

- 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/gnu/flex/
- E2fsprogs (1.18): ftp://tsx−11.mit.edu/pub/linux/packages/ext2fs/
- File (3.26): http://www.linuxfromscratch.org/download/file−3.26−lfs.tar.gz
- Groff (1.15): ftp://ftp.gnu.org/gnu/groff/
- Ld.so (1.9.9): ftp://tsx−11.mit.edu/pub/linux/packages/GCC/
- Libtool (1.3.4): ftp://ftp.gnu.org/gnu/libtool/
- Linux86 (0.14.3): http://www.linuxfromscratch.org/download/linux86−0.14.3−lfs.tar.gz
- Lilo (21.4.2): ftp://sd.dynhost.com/pub/linux/lilo
- Shadow Password Suite (19990827): ftp://piast.t19.pwr.wroc.pl/pub/linux/shadow/
- Man (1.5h1): ftp://ftp.win.tue.nl/pub/linux−local/utils/man/
- Modutils (2.3.9): ftp://ftp.ocs.com.au/pub/modutils/
- Procinfo (17): ftp://ftp.cistron.nl/pub/people/svm/
- Procps (2.0.6): ftp://people.redhat.com/johnsonm/procps/
- Psmisc (19): ftp://lrcftp.epfl.ch/pub/linux/local/psmisc/
- Start−stop−daemon (0.4.1): http://www.linuxfromscratch.org/download/ssd−0.4.1.tar.gz
-

Sysklogd (1.3.31): ftp://sunsite.unc.edu/pub/Linux/system/daemons/

- Vim−rt + Vim−src (5.6): ftp://ftp.vim.org/pub/editors/vim/unix/

Chapter 4. Preparing a new partition

How we are going to do things

We are going to build the LFS system 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 safe yourself a lot of trouble if your normal system uses glibc−2.0 or newer. Libc5 isn't supported by this book, though it isn't impossible to use a libc5 system if you have no choice.

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 at least 5 00 MB. You can get away with around 250MB for a bare system with no extra bells and whistles (such as software for emailing, networking, Internet, X Window System and such). If you already have a Linux Native partition available, you can skip this subsection.

Start the fdisk program (or some other 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 that 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 hda5 (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 was hda5, you would run:

```
root:~# mke2fs /dev/hda5
```

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 date to it) is mounting it. If you mount it under /mnt/hda5, you can access this partition by going to the /mnt/hda5 directory and then do whatever you need to do. This document will assume that you have mounted the partition on a subdirectory under /mnt. It doesn't matter which directory you choose (or you can use just the /mnt directory as the mount point), but a good practice is to create a directory with the same name as the partition's designation. In my case the LFS partition is called hda5 and therefore I mount it on /mnt/hda5

Create the /mnt directory if it doesn't exist yet by runnning:

```
root:~# mkdir /mnt
```

Create the /mnt/xxx directory by running:

```
root:~# mkdir /mnt/xxx
```

Replace "xxx" by your partition's designation.

Now mount the LFS partition by running:

```
root:~# mount /dev/xxx  /mnt/xxx
```

Replace "xxx" by your partition's designation.

This directory (/mnt/xxx) 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/xxx/etc" where "xxx" is replaced by your partition's designation.

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:

```
root:~#   cd $LFS
root:/mnt/hda5#   mkdir bin boot dev etc home lib mnt proc
root \
> sbin tmp usr var
root:/mnt/hda5#   cd $LFS/usr
root:/mnt/hda5/usr#   mkdir bin include lib local sbin share
src
root:/mnt/hda5/usr#   ln -s share/man man
root:/mnt/hda5/usr#   ln -s share/doc doc
root:/mnt/hda5/usr#   ln -s share/info info
root:/mnt/hda5/usr#   ln -s ../etc etc
root:/mnt/hda5/usr#   ln -s ../var var
root:/mnt/hda5/usr#   cd $LFS/usr/share
root:/mnt/hda5/usr/share#   mkdir dict doc info locale man
nls \
> misc terminfo zoneinfo
root:/mnt/hda5/usr/share#   cd $LFS/usr/share/man
root:/mnt/hda5/usr/share/man#   mkdir man1 man2 man3 man4
man5 \
> man6 man7 man8
root:/mnt/hda5/usr/share/man#   cd $LFS/var
root:/mnt/hda5/var#   mkdir lock log run spool 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).

Copying the /dev directory

We can create every single file that we need to be in the $LFS/dev directory using the mknod command, but that just takes up a lot of time. I choose to just simply copy the current /dev directory to the $LFS partition. Use this command to copy the entire directory while preserving original rights, symlinks and ownerships:

```
root:~# cp −av /dev $LFS
root:~#   chown root $LFS/dev/*
```

I'm aware that this isn't the best way to create the files. I know of a MAKEDEV script but I choose not to use it. I'm actually waiting for the 2.4 Linux kernel to be released. The kernel has a stable version of the devfs which this book will use in the future. Devfs is a dynamic file system which makes the static files in /dev obsolete. You mount the dev file system to a mount point (kind of like the way the proc file system works) and the kernel will create the files in /dev you need on−the−fly. So the waiting is for the next stable kernel to be released.

Chapter 5. Installing basic system software

In this chapter 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 optional issues such as setting up networking, Internet servers + clients (telnet, ftp, http, email), setting up Internet itself and the X Window System. You can skip chapters at your own discretion. If you don't plan on going online with the LFS system there's little use to setup Internet for example.

This chapter is devided in two chunks. The first part installs a few necessary programs on the LFS system. These programs are needed to install the rest of the programs that belong to a basic system. When the first part is done, we will enter a chroot'ed environment. This means that we start a shell with $LFS as the root directory (instead of the usual / directory as the root directory). This has the same effect as rebooting the computer into the LFS system, but this way we don't have to reboot. If something goes wrong, you don't need to reboot back in the normal Linux system to fix whatever you need to fix. You just open a new shell on a virtual console, or start a new xterm and you can do what you need to do.

The software in the first part will be linked statically. These programs will be re−installed in the second part and linked dynamically. The reason for the static version first is that there is a chance that our normal Linux system and our LFS system−to−be don't use the same C Library versions. If the programs in the first part are linked against an older C library version, those program might not work too well on the LFS system.

About debugging symbols

Every program and library is by default compiled with debugging symbols. This means you can run a program or library through a debugger and the debugger's output will be more user friendly. These debugging symbols also enlarge the program or library significantly. This document will not install software without debugging symbols (as I don't know if the majority of readers do or do not debug software). In stead, you can remove those symbols manually if you want with the strip program.

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/*).

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

- A static Bash binary with debugging symbols: 2.3MB
- A static Bash binary without debugging symbols: 645KB
- A dynamic Bash binary with debugging symbols: 1.2MB
- A dynamic Bash binary without debugging symbols: 478KB
- $LFS/lib and $LFS/usr/lib (glibc and gcc files) with debugging symbols: 87MB
- $LFS/lib and $LFS/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. The difference would be even greater when I would do this at the end of this book when everything is installed.

Preparing the LFS system for installing basic system software

Installing Bash

Install Bash by running the following commands:

```
root:/mnt/hda5/usr/src/bash-2.04#   ./configure
--enable-static-link
root:/mnt/hda5/usr/src/bash-2.04#   make
root:/mnt/hda5/usr/src/bash-2.04#   make -e prefix=$LFS/usr
install
root:/mnt/hda5/usr/src/bash-2.04#   mv $LFS/usr/bin/bash
$LFS/bin
root:/mnt/hda5/usr/src/bash-2.04#   cd $LFS/bin
root:/mnt/hda5/bin#   ln -s bash sh
```

Installing Binutils

Install Binutils by running the following commands:

```
root:/mnt/hda5/usr/src/binutils-2.9.5.0.37#   ./configure \
> --prefix=/usr
root:/mnt/hda5/usr/src/binutils-2.9.5.0.37#   make -e \
> LDFLAGS=-all-static
root:/mnt/hda5/usr/src/binutils-2.9.5.0.37a#   make -e \
> prefix=$LFS/usr install
```

Installing Bzip2

Before we can install Bzip2 we need to modify the `Makefile` file. Open the `Makefile` file in a text editor and find the lines that start with *$(CC) $(CFLAGS) -o*

Replace those parts with: *$(CC) $(CFLAGS) $(LDFLAGS) -o*

Now install Bzip2 by running the following commands:

```
root:/mnt/hda5/usr/src/bzip2-0.9.5d#   make -e LDFLAGS=-static
root:/mnt/hda5/usr/src/bzip2-0.9.5d#   make -e
PREFIX=$LFS/usr \
> install
root:/mnt/hda5/usr/src/bzip2-0.9.5d#   cd $LFS/usr/bin
root:/mnt/hda5/usr/bin#   mv bunzip2 bzip2 $LFS/bin
```

Installing Diffutils

Install Diffutils by running the following commands:

```
root:/mnt/hda5/usr/src/diffutils-2.7#   ./configure
--prefix=/usr
root:/mnt/hda5/usr/src/diffutils-2.7#   make -e
LDFLAGS=-static
root:/mnt/hda5/usr/src/diffutils-2.7#   make -e
prefix=$LFS/usr \
> install
```

This package is known to cause static link problems on certain platforms. If you're having trouble compiling this package as well, you can download a patch from http://www.linuxfromscratch.org/download/diffutils–2.7.patch.gz

Install this patch by running the following command:

```
root:/mnt/hda5/usr/src/diffutils-2.7#   patch -Np1 \
> -i ../diffutils-2.7.patch
```

Now recompile the package using the same commands as above.

Installing Fileutils

Install Fileutils by running the following commands:

```
root:/mnt/hda5/usr/src/fileutils-4.0#   ./configure \
> --disable-nls --prefix=/usr
root:/mnt/hda5/usr/src/fileutils-4.0#   make -e
LDFLAGS=-static
root:/mnt/hda5/usr/src/fileutils-4.0#   make -e
prefix=$LFS/usr \
> install
root:/mnt/hda5/usr/src/fileutils-4.0#   cd $LFS/usr/bin
root:/mnt/hda5/usr/bin#   mv chgrp chmod chown cp dd df ln
$LFS/bin
root:/mnt/hda5/usr/bin#   mv ls mkdir mknod mv rm rmdir sync
$LFS/bin
```

Installing 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. Many glibc−2.0 based systems have gcc−2.7.2.3 installed and you can't compile glibc−2.1.3 with that compiler. Many glibc−2.1 based systems have egcs−2.95.x installed and that version doesn't work too well either (sometimes it works fine, sometimes it doesn't depending on various circumstances).

If you normal Linux system does not have gcc−2.95.2 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.

Install GCC by running the following commands:

```
root:/mnt/hda5/usr/src#   mkdir $LFS/usr/src/gcc-build
root:/mnt/hda5/usr/src#   cd $LFS/usr/src/gcc-build
root:/mnt/hda5/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++
root:/mnt/hda5/usr/src/gcc-build#   make bootstrap
root:/mnt/hda5/usr/src/gcc-build#   make install
```

Installing GCC on the LFS system

Install GCC by running the following commands:

```
root:/mnt/hda5/usr/src#   mkdir $LFS/usr/src/gcc-build
root:/mnt/hda5/usr/src#   cd $LFS/usr/src/gcc-build
root:/mnt/hda5/usr/src/gcc-build#   ../gcc-2.95.2/configure \
> --prefix=/usr --with-local-prefix=/usr \
> --with-gxx-include-dir=/usr/include/g++ \
> --enable-languages=c,c++ --disable-nls
root:/mnt/hda5/usr/src/gcc-build#   make -e LDFLAGS=-static
bootstrap
root:/mnt/hda5/usr/src/gcc-build#   make -e prefix=$LFS/usr \
> local_prefix=$LFS/usr gxx_include_dir=$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:

```
root:~#   cd $LFS/lib
root:/mnt/hda5/lib#   ln -s
../usr/lib/gcc-lib/<host>/2.95.2/cpp cpp
root:/mnt/hda5/lib#   cd $LFS/usr/lib
root:/mnt/hda5/usr/lib#   ln -s gcc-lib/<host>/2.95.2/cpp cpp
root:/mnt/hda5/usr/lib#   cd $LFS/usr/bin
root:/mnt/hda5/usr/bin#   ln -s gcc cc
```

Replace <host> with the directory where the gcc−2.95.2 files are installed (which is i686−unknown−linux in my case).

Installing Glibc

A note on the glibc−crypt package

An excerpt from the README file that is distributed with the glibc−crypt package:

The add−on is not included in the main distribution of the GNU C library because some governments, most notably those of France, Russia, and the US, have very restrictive rules governing the distribution and use of encryption software. Please read the node "Legal Problems" in the manual for more details.

In particular, the US does not allow export of this software without a licence, including via the Internet. So please do not download it from the main FSF FTP site at ftp.gnu.org if you are outside the US. This software was completely developed outside the US.

"This software" refers to the glibc−crypt package at ftp://ftp.gwdg.de/pub/linux/glibc/. This law only affects people who don't live in the US. It's not prohibited to import DES software, so if you live in the US you can import the file safely from Germany without breaking cryptographic laws. This law is changing lately and I don't know what the status of it is at the moment. Better be safe than sory.

Installing Glibc

Copy the Glibc−crypt and Glibc−linuxthreads archives into the unpacked glibc directory

Unpack the glibc−crypt and glibc−linuxthreads archives there, but don't enter the created directories. Just unpack 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 in that directory you create a new file `configparms` containing:

```
# Begin configparms

slibdir=/lib
sysconfdir=/etc

# End configparms
```

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

```
root:/mnt/hda5/usr/src/glibc-build#
../glibc-2.1.3/configure \
> --prefix=/usr --enable-add-ons
root:/mnt/hda5/usr/src/glibc-build#   make
root:/mnt/hda5/usr/src/glibc-build#   make install_root=$LFS
install
```

Change to the `$LFS/usr/src/glibc-build` directory and install Glibc by running the following command if your system did not already have a suitable GCC version installed and you just installed GCC−2.95.2 on your normal Linux system a little while ago:

```
root:/mnt/hda5/usr/src/glibc-build#
CC=/usr/local/gcc2952/bin/gcc \
> ../glibc-2.1.3/configure --prefix=/usr \
> --enable-add-ons
root:/mnt/hda5/usr/src/glibc-build#   make
root:/mnt/hda5/usr/src/glibc-build#   make install_root=$LFS
install
```

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,userid's and groupid's. You can check which C library version your normal Linux system uses by running:

```
root:~# ls /lib/libc*
```

Your system uses glib−2.0 if there is a file that looks like *libc−2.0.7.so*

Your system uses glibc−2.1 if there is a file that looks like *libc−2.1.3.so*

Of course, the micro version number can be different (you could have libc−2.1.2 or libc−2.1.1 for example).

If you have a libc−2.0.x file copy the NSS library files by running:

```
root:~# cp −av /lib/libnss*  $LFS/lib
```

There are a few distributions that don't have files from which you can see which version of the C Library it is. If that's the case, it will be hard to determine which C library version you exactly have. Try to obtain this information using your distribution's installation tool. It often says which version it has available. If you can't figure out at all which C Library version is used, then copy the NSS files anyway and hope for the best. That's the best advise I can give I'm afraid.

Installing Grep

Install Grep by running the following commands:

```
root:/mnt/hda5/usr/src/grep-2.4.2#   ./configure
--prefix=/usr \
> --disable-nls
root:/mnt/hda5/usr/src/grep-2.4.2#   make -e LDFLAGS=-static
root:/mnt/hda5/usr/src/grep-2.4.2#   make -e prefix=$LFS/usr
install
```

This package is known to cause static linking problems on certain platforms. If you're having trouble compiling this package as well, you can download a patch from http://www.linuxfromscratch.org/download/grep–2.4.2.patch.gz

Install this patch by running the following command:

```
root:/mnt/hda5/usr/src/grep-2.4.2#    patch -Np1 \
> -i ../grep-2.4.2.patch
```

Now recompile the package using the same commands as above.

Installing Gzip

Install Gzip by running the following commands:

```
root:/mnt/hda5/usr/src/gzip-1.2.4a#   ./configure
--prefix=/usr
root:/mnt/hda5/usr/src/gzip-1.2.4a#   make -e LDFLAGS=-static
root:/mnt/hda5/usr/src/gzip-1.2.4a#   make -e prefix=$LFS/usr
install
root:/mnt/hda5/usr/src/gzip-1.2.4a#   cd $LFS/usr/bin
root:/mnt/hda5/usr/bin#   mv gunzip gzip $LFS/bin
```

This package is known to cause compilation problems on certain platforms. If you're having trouble compiling this package as well, you can download a fixed package from http://www.linuxfromscratch.org/download/gzip–1.2.4a.patch.gz

Install this patch by running the following command:

```
root:/usr/src/#    patch -Np1 \
> -i ../findutils-4.1.patch.gz
```

Now recompile the package using the same commands as above.

Installing Make

Install Make by running the following commands:

```
root:/mnt/hda5/usr/src/make-3.78.1#   ./configure
--prefix=/usr
root:/mnt/hda5/usr/src/make-3.78.1#   make -e LDFLAGS=-static
root:/mnt/hda5/usr/src/make-3.78.1#   make -e prefix=$LFS/usr
install
```

Installing Sed

Install Sed by running the following commands:

```
root:/mnt/hda5/usr/src/sed-3.02#   ./configure --prefix=/usr
root:/mnt/hda5/usr/src/sed-3.02#   make -e LDFLAGS=-static
root:/mnt/hda5/usr/src/sed-3.02#   make -e prefix=$LFS/usr
install
root:/mnt/hda5/usr/src/sed-3.02#   mv $LFS/usr/bin/sed
$LFS/bin
```

This package is known to cause static linking problems on certain platforms. If you're having trouble compiling this package as well, you can download a patch from http://www.linuxfromscratch.org/download/sed–3.02.patch.gz

Install this patch by running the following command:

```
root:/usr/src/sed-3.02#   patch -Np1 \
```

```
> -i ../sed-3.02.patch.gz
```

Now recompile the package using the same commands as above.

Installing Shellutils

Install Shellutils by running the following commands:

```
root:/mnt/hda5/usr/src/sh-utils-2.0#   ./configure
--prefix=/usr \
> --disable-nls
root:/mnt/hda5/usr/src/sh-utils-2.0#   make -e LDFLAGS=-static
root:/mnt/hda5/usr/src/sh-utils-2.0#   make -e
prefix=$LFS/usr \
> install
root:/mnt/hda5/usr/src/sh-utils-2.0#   cd $LFS/usr/bin
root:/mnt/hda5/usr/bin#   mv date echo false pwd stty $LFS/bin
root:/mnt/hda5/usr/bin#   mv su true uname hostname $LFS/bin
```

Installing Tar

Install Tar by running the following commands:

```
root:/mnt/hda5/usr/src/tar-1.13#   ./configure --prefix=/usr \
> --disable-nls
root:/mnt/hda5/usr/src/tar-1.13#   make -e LDFLAGS=-static
root:/mnt/hda5/usr/src/tar-1.13#   make -e prefix=$LFS/usr
install
root:/mnt/hda5/usr/src/tar-1.13#   mv $LFS/usr/bin/tar
$LFS/bin
```

Installing Textutils

Install Textutils by running the following commands:

```
root:/mnt/hda5/usr/src/textutils-2.0#   ./configure
--prefix=/usr \
> --disable-nls
root:/mnt/hda5/usr/src/textutils-2.0#   make -e
LDFLAGS=-static
root:/mnt/hda5/usr/src/textutils-2.0#   make -e
prefix=$LFS/usr \
> install
root:/mnt/hda5/usr/src/textutils-2.0#   mv $LFS/usr/bin/cat
$LFS/bin
```

Creating passwd and group files

Create a new file `$LFS/etc/passwd` containing the following:

```
# Begin /etc/passwd
root:kRKb4s/LH8yDQ:0:0:root:/root:/bin/bash
# End /etc/passwd
```

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

```
# Begin /etc/group
root::0:
# End /etc/group
```

The encoded password above is: *lfs123*

Installing basic system software

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.

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

Entering the chroot'ed environment

Enter the following commands to setup 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/xxx but the shell doesn't know that).

```
root:~#   cd $LFS
root:/mnt/hda5#   chroot $LFS bash
```

Installing 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 setup the proper symlinks in /usr/include. Create the `/usr/include/linux` and `/usr/include/asm` symlinks by running the following commands:

```
root:~#   cd /usr/include
root:/usr/include#   ln -s ../src/linux/include/linux linux
root:/usr/include#   ln -s ../src/linux/include/asm-i386 asm
```

Installing Ed

Install Ed by running the following commands:

```
root:/usr/src/ed-0.2#   ./configure --prefix=/usr
root:/usr/src/ed-0.2#   make
root:/usr/src/ed-0.2#   make install
```

Installing Patch

Install Patch by running the following commands:

```
root:/usr/src/patch-2.5.4#   ./configure --prefix=/usr
root:/usr/src/patch-2.5.4#   make
root:/usr/src/patch-2.5.4#   make install
```

Installing GCC

Install GCC by running the following commands:

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

Installing Bison

Install Bison by running the following commands:

```
root:/usr/src/bison-1.28#   ./configure --prefix=/usr \
> --datadir=/usr/share/bison
root:/usr/src/bison-1.28#   make
root:/usr/src/bison-1.28#   make install
```

Installing Mawk

Install Mawk by running the following commands:

```
root:/usr/src/mawk-1.3.3#   ./configure
root:/usr/src/mawk-1.3.3#   make
root:/usr/src/mawk-1.3.3#   make -e BINDIR=/usr/bin \
> MANDIR=/usr/share/man/man1 install
root:/usr/src/mawk-1.3.3#   cd /usr/bin
root:/usr/bin#   ln -s mawk awk
```

Installing Findutils

Install Findutils by running the following commands:

```
root:/usr/src/findutils-4.1#   ./configure --prefix=/usr
root:/usr/src/findutils-4.1#   make
root:/usr/src/findutils-4.1#   make install
```

This package is known to cause compilation problem. If you're having trouble compiling this package as well, you can download a patch from http://www.linuxfromscratch.org/download/findutils–4.1.patch.gz

Install this patch by running the following command:

```
root:/usr/src/#    patch -Np1 \
> -i ../findutils-4.1.patch.gz
```

Now recompile the package using the same commands as above.

Installing Termcap

Install Termcap by running the following commands:

```
root:/usr/src/termcap-1.3#   ./configure --prefix=/usr
root:/usr/src/termcap-1.3#   make
root:/usr/src/termcap-1.3#   make install
```

Installing Ncurses

Install Ncurses by running the following commands:

```
root:/usr/src/ncurses-5.0#   ./configure --prefix=/usr
--with-shared
root:/usr/src/ncurses-5.0#   make
root:/usr/src/ncurses-5.0#   make install
```

Installing Less

Install Less by running the following commands:

```
root:/usr/src/less-340#   ./configure --prefix=/usr
root:/usr/src/less-340#   make
root:/usr/src/less-340#   make install
root:/usr/src/less-340#   mv /usr/bin/less /bin
```

Installing Perl

Install Perl by running the following commands:

```
root:/usr/src/perl-5.6.0#   ./Configure
root:/usr/src/perl-5.6.0#   make
root:/usr/src/perl-5.6.0#   make test
root:/usr/src/perl-5.6.0#   make install
```

Note that you have to change the installation path to `/usr` yourself. The Perl installation defaults to the /usr/local/subdir

Also note that a few tests during the make test phase will fail because we don't have network support installed yet.

Installing M4

Install M4 by running the following commands:

```
root:/usr/src/m4-1.4#   ./configure --prefix=/usr
root:/usr/src/m4-1.4#   make
root:/usr/src/m4-1.4#   make install
```

Installing Texinfo

Install Texinfo by running the following commands:

```
root:/usr/src/texinfo-4.0#   ./configure --prefix=/usr
root:/usr/src/texinfo-4.0#   make
root:/usr/src/texinfo-4.0#   make install
```

Installing Autoconf

Install Autoconf by running the following commands:

```
root:/usr/src/autoconf-2.13#   ./configure --prefix=/usr
root:/usr/src/autoconf-2.13#   make
root:/usr/src/autoconf-2.13#   make install
```

Installing Automake

Install Automake by running the following commands:

```
root:/usr/src/automake-1.4#   ./configure --prefix=/usr
root:/usr/src/automake-1.4#   make install
```

Installing Bash

Install Bash by running the following commands:

```
root:/usr/src/bash-2.04#   ./configure --prefix=/usr
root:/usr/src/bash-2.04#   make
root:/usr/src/bash-2.04#   make install
root:/usr/src/bash-2.04#   mv /usr/bin/bash /bin
```

Installing Flex

Install Flex by running the following commands:

```
root:/usr/src/flex-2.5.4a#   ./configure --prefix=/usr
root:/usr/src/flex-2.5.4a#   make
root:/usr/src/flex-2.5.4a#   make install
```

Installing Binutils

Install Binutils by running the following commands:

```
root:/usr/src/binutils-2.9.5.0.37#   ./configure --prefix=/usr
root:/usr/src/binutils-2.9.5.0.37#   make
```

```
root:/usr/src/binutils-2.9.5.0.37#   make install
```

Installing Bzip2

Install Bzip2 by running the following commands:

```
root:/usr/src/bzip2-0.9.5d#   make
root:/usr/src/bzip2-0.9.5d#   make PREFIX=/usr install
root:/usr/src/bzip2-0.9.5d#   cd /usr/bin
root:/usr/bin#   mv bunzip2 bzip2 /bin
```

Installing Diffuitls

Install Diffutils by running the following commands:

```
root:/usr/src/diffutils-2.7#   ./configure --prefix=/usr
root:/usr/src/diffutils-2.7#   make
root:/usr/src/diffutils-2.7#   make install
```

Installing E2fsprogs

Install E2fsprogs by running the following commands:

```
root:/usr/src/e2fsprogs-1.18#   ./configure --prefix=/usr
root:/usr/src/e2fsprogs-1.18#   make
root:/usr/src/e2fsprogs-1.18#   make install
root:/usr/src/e2fsprogs-1.18#   cd /usr/sbin
root:/usr/sbin#   mv *e2* *fs* mklost+found /sbin
```

Installing File

Install File by running the following commands:

```
root:/usr/src/file-3.26#  ./configure --prefix=/usr
root:/usr/src/file-3.26#  make
root:/usr/src/file-3.26#  make install
```

Installing Fileutils

Install Fileutils by running the following commands:

```
root:/usr/src/fileutils-4.0#  ./configure --prefix=/usr
root:/usr/src/fileutils-4.0#  make
root:/usr/src/fileutils-4.0#  make install
root:/usr/src/fileutils-4.0#  cd /usr/bin
root:/usr/bin#  mv chgrp chmod chown cp dd df ln /bin
root:/usr/bin#  mv ls mkdir mknod mv rm rmdir sync /bin
```

Installing Grep

Install Grep by running the following commands:

```
root:/usr/src/grep-2.4.2#  ./configure --prefix=/usr
root:/usr/src/grep-2.4.2#  make
root:/usr/src/grep-2.4.2#  make install
```

Installing Groff

Install Groff by running the following commands:

```
root:/usr/src/groff-1.15#   ./configure --prefix=/usr
root:/usr/src/groff-1.15#   make
root:/usr/src/groff-1.15#   make install
```

Installing Gzip

Install Gzip by running the following commands:

```
root:/usr/src/gzip-1.2.4a#   ./configure --prefix=/usr
root:/usr/src/gzip-1.2.4a#   make
root:/usr/src/gzip-1.2.4a#   make install
root:/usr/src/gzip-1.2.4a#   cd /usr/bin
root:/usr/bin#   mv gunzip gzip /bin
```

Installing Ld.so

Install Ld.so by running the following commands:

```
root:/usr/src/ld.so-1.9.10#   cd util
root:/usr/src/ld.so-1.9.10/util#   make ldd ldconfig
root:/usr/src/ld.so-1.9.10/util#   cp ldd /bin
root:/usr/src/ld.so-1.9.10/util#   cp ldconfig /sbin
root:/usr/src/ld.so-1.9.10/util#   rm /usr/bin/ldd
```

Installing Libtool

Install Libtool by running the following commands:

```
root:/usr/src/libtool-1.3.4#   ./configure --prefix=/usr
root:/usr/src/libtool-1.3.4#   make
root:/usr/src/libtool-1.3.4#   make install
```

Installing Linux86

Install Linux86 by running the following commands:

```
root:/usr/src/linux-86#   cd as
root:/usr/src/linux-86/as#   make
root:/usr/src/linux-86#   make install
root:/usr/src/linux-86#   cd ../ld
root:/usr/src/linux-86#   make ld86
root:/usr/src/linux-86#   make install
```

Installing Lilo

Install Lilo by running the following commands:

```
root:/usr/src/lilo-21.4.1#   make
root:/usr/src/lilo-21.4.1#   make install
```

Installing Make

Install Make by running the following commands:

```
root:/usr/src/make-3.78.1#   ./configure --prefix=/usr
root:/usr/src/make-3.78.1#   make
root:/usr/src/make-3.78.1#   make install
```

Installing Shell Utils

Install Shellutils by running the following commands:

```
root:/usr/src/sh-utils-2.0#   ./configure --prefix=/usr
root:/usr/src/sh-utils-2.0#   make
root:/usr/src/sh-utils-2.0#   make install
root:/usr/src/sh-utils-2.0#   cd /usr/bin
root:/usr/bin#   mv date echo false pwd stty /bin
root:/usr/bin#   mv su true uname hostname /bin
```

Installing Shadow Password Suite

Install the Shadow Password Suite by running the following commands:

```
root:/usr/src/shadow-19990827#   ./configure --prefix=/usr
root:/usr/src/shadow-19990827#   make
root:/usr/src/shadow-19990827#   make install
root:/usr/src/shadow-19990827#   cd etc
root:/usr/src/shadow-19990827/etc#   cp limits login.access \
> login.defs.linux shells suauth /etc
root:/usr/src/shadow-19990827#   mv /etc/login.defs.linux \
> /etc/login.defs
```

Installing Man

Install Man by running the following commands:

```
root:/usr/src/man-1.5h1#   ./configure -default
root:/usr/src/man-1.5h1#   make
root:/usr/src/man-1.5h1#   make install
```

Installing Modutils

Install Modutils by running the following commands:

```
root:/usr/src/modutils-2.3.9#   ./configure
root:/usr/src/modutils-2.3.9#   make
root:/usr/src/modutils-2.3.9#   make install
```

Installing Procinfo

Install Procinfo by running the following commands:

```
root:/usr/src/procinfo-17#   make
root:/usr/src/procinfo-17#   make install
```

Installing Procps

Install Procps by running the following commands:

```
root:/usr/src/procps-2.0.6#   gcc -O3 -Wall -Wno-unused -c
watch.c
root:/usr/src/procps-2.0.6#   make
root:/usr/src/procps-2.0.6#   make -e XSCPT="" install
root:/usr/src/procinfo-17#   mv /usr/bin/kill /bin
```

Installing Psmisc

Install Psmisc by running the following commands:

```
root:/usr/src/psmisc-19#   make
root:/usr/src/psmisc-19#   make install
```

Installing Sed

Install Sed by running the following commands:

```
root:/usr/src/sec-3.02#   ./configure --prefix=/usr
root:/usr/src/sec-3.02#   make
root:/usr/src/sec-3.02#   make install
root:/usr/src/sec-3.02#   mv /usr/bin/sed /bin
```

Installing Start–stop–daemon

Install Start–stop–daemon by running the following commands:

```
root:/usr/src/ssd-0.4.1#   make
root:/usr/src/ssd-0.4.1#   make install
```

Installing Sysklogd

Install Sysklogd by running the following commands:

```
root:/usr/src/sysklogd-1.3-31#   make
root:/usr/src/sysklogd-1.3-31#   make install
```

Installing Sysvinit

Install Sysvinit by running the following commands:

```
root:/usr/src/sysvinit-2.78#   cd src
root:/usr/src/sysvinit-2.78#   make
root:/usr/src/sysvinit-2.78#   make install
```

Installing Tar

Install Tar by running the following commands:

```
root:/usr/src/tar-1.13#   ./configure --prefix=/usr
root:/usr/src/tar-1.13#   make
root:/usr/src/tar-1.13#   make install
root:/usr/src/tar-1.13#   mv /usr/bin/tar /bin
```

Installing Textutils

Install Textutuils by running the following commands:

```
root:/usr/src/textutils-2.0#   ./configure --prefix=/usr
root:/usr/src/textutils-2.0#   make
root:/usr/src/textutils-2.0#   make install
root:/usr/src/textutils-2.0#   mv /usr/bin/cat /bin
```

Installing Vim

You need to unpack both the vim-rt and vim-src packages to install Vim. Install Vim by running the following commands:

```
root:/usr/src/vim-5.6#   ./configure --prefix=/usr
root:/usr/src/vim-5.6#   make
root:/usr/src/vim-5.6#   make install
root:/usr/src/vim-5.6#   cd /usr/bin
root:/usr/bin#   ln -s vim vi
```

Installing Util−Linux

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

HAVE_PASSWD=yes
HAVE_SLN=yes
HAVE_TSORT=yes

Install Util−Linux by running the following commands:

```
root:/usr/src/util-linux-2.10h#   groupadd -g 5 tty
root:/usr/src/util-linux-2.10h#   ./configure
root:/usr/src/util-linux-2.10h#   make
root:/usr/src/util-linux-2.10h#   make install
```

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:

```
root:~#   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 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 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` containing:

```
# Begin /etc/nsswitch.conf

passwd: files
group: files
shadow: 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
```

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:

```
root:~#   cd /etc
root:/etc#   rm localtime
root:/etc#   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 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.

```
root:~#   cp /etc/lilo.conf $LFS/etc
root:~#   cp /boot/* $LFS/boot
```

If your normal Linux system does not have (all of) it's kernel images in /boot, then check your /etc/lilo.conf file for the location of those files and copy those as well to the location where /etc/lilo.conf expects them to be. Or you can copy them to /boot regardless and modify the /etc/lilo.conf file so it contains the new paths for the images as you have them on the LFS system. Either way works fine, it's up to you how you want to do it.

Configuring Sysklogd

Create the `/etc/syslog.conf` file containing the following:

```
# Begin /etc/syslog.conf

auth,authpriv.* –/var/log/auth.log
*.*;auth,authpriv.none –/var/log/sys.log
daemon.* –/var/log/daemon.log
kern.* –/var/log/kern.log
mail.* –/var/log/mail.log
user.* –/var/log/user.log
*.emerg *

# End /etc/syslog.conf
```

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. 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 shadowed passwords.

If you decide you don't want to use shadowed passwords (after you're read the doc/HOWTO document), you still use this archive since the utilities in this archive are also used on system which have shadowed passwords disabled. You can read all about this in the HOWTO. Also note that you can switch between shadow and non−shadow at any point you want.

Now is a very good moment to read chapter 5 of the doc/HOWTO file. You can read how you can test if shadowing works and if not, how to disable it. If it doesn't work and you haven't tested it, you'll end up with an unusable system after you logout of all your consoles, since you won't be able to login anymore. You can easily fix this by passing the init=/sbin/sulogin parameter to the kernel, unpack the util−linux archive, go to the login−utils directory, build the login program and replace the /bin/login by the one in the util−linux package. Things are never hopelessly messed up (at least not under Linux), but you can avoid a hassle by testing properly and reading manuals ;)

Configuring Sysvinit

Create a new file `/etc/inittab` containing the following:

```
# Begin /etc/inittab

id:2:initdefault:

si::sysinit:/etc/init.d/rcS

su:S:wait:/sbin/sulogin

l0:0:wait:/etc/init.d/rc 0
l1:1: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

ft:6:respawn:/sbin/sulogin

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

```
1:2345:respawn:/sbin/agetty /dev/tty1 9600
2:2345:respawn:/sbin/agetty /dev/tty2 9600
3:2345:respawn:/sbin/agetty /dev/tty3 9600
4:2345:respawn:/sbin/agetty /dev/tty4 9600
5:2345:respawn:/sbin/agetty /dev/tty5 9600
6:2345:respawn:/sbin/agetty /dev/tty6 9600

# End /etc/inittab
```

Creating the /var/run/utmp file

Programs like login, shutdown, uptime and others want to read from and write to the /var/run/utmp file. This file contains information about who is currently logged in. It also contains information on when the computer was last booted and shutdown.

Create the `/var/run/utmp` and give it the proper permissions by running the following commands:

```
root:~# touch /var/run/utmp
root:~# chmod 644 /var/run/utmp
```

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` containing the following:

```
set nocompatible
set bs=2
```

Chapter 6. Creating system boot scripts

These bootscripts are started at system boot time. The scripts are responsible for mounting the root file system in read−write mode, activating swap, setting up some system settings and starting the various daemons that our system needs.

Preparing the directories and master files

You need the Sysvinit package again for this section.

We need to start by creating a few extra directories that are used by the boot scripts. Create these directories by running:

```
root:~#   cd /etc
root:/etc#   mkdir rc0.d rc1.d rc2.d rc3.d
root:/etc#   mkdir rc4.d rc5.d rc6.d init.d rcS.d
```

Go to the unpacked Sysvinit source directory and copy the `Debian/etc/init.d/rc` file to `/etc/init.d` by running:

```
root:~#   cp /usr/src/sysvinit-2.78/debian/etc/init.d/rc
/etc/init.d
```

The second boot script is called `rcS` and we'll create that one from scratch. Create a new file `/etc/init.d/rcS` containing the following:

```
#!/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
```

Creating the reboot script

Create a new file `/etc/init.d/reboot` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/reboot

echo "System reboot in progress..."

/sbin/reboot -d -f -i

# End /etc/init.d/reboot
```

Creating the halt script

Create a new file `/etc/init.d/halt` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/halt

/sbin/halt –d –f –i –p

# End /etc/init.d/halt
```

Creating the mountfs script

Create a new file `/etc/init.d/mountfs` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/mountfs

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

echo -n "Remounting root file system in read-write mode..."
/bin/mount -n -o remount,rw /
check_status

echo > /etc/mtab
/bin/mount -f -o remount,rw /

echo -n "Mounting proc file system..."
/bin/mount proc
check_status

# End /etc/init.d/mountfs
```

Creating the umountfs script

Create a new file `/etc/init.d/umountfs` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/umountfs

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

echo -n "Deactivating swap..."
/sbin/swapoff -a
check_status

echo -n "Unmounting file systems..."
/bin/umount -a -r
check_status

# End /etc/init.d/umountfs
```

Creating the sendsignals script

Create a new file `/etc/init.d/sendsignals` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/sendsignals

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

echo −n "Sending all processes the TERM signal..."
/sbin/killall5 −15
check_status

echo −n "Sending all processes the KILL signal..."
/sbin/killall5 −9
check_status

# End /etc/init.d/sendsignals
```

Creating the checkroot script

Create a new file `/etc/init.d/checkroot` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/checkroot

echo −n "Activating swap..."
/sbin/swapon −a

if [ −f /fastboot ]
then
  echo "Fast boot, no file system check"
else
  /bin/mount −n −o remount,ro /
  if [ $? = 0 ]
  then
    if [ −f /forcecheck ]
    then
      force="−f"
    else
      force=""
    fi

    echo "Checking root file system..."
    /sbin/fsck $force −a /

    if [ $? −gt 1 ]
    then
      echo
      echo "fsck failed. Please repair your file system manually by"
      echo "running fsck without the −a option"
      echo
      echo "Please note that the file system is currently mounted in"
      echo "read−only mode."
      echo
      echo "I will start sulogin now. CTRL+D will reboot your system."
      /sbin/sulogin
      /sbin/reboot −f
    fi
  else
    echo "Cannot check root file system because it is not mounted in"
    echo "read−only mode."
  fi
fi

# End /etc/init.d/checkroot
```

Creating the sysklogd script

Create a new file `/etc/init.d/sysklogd` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/sysklogd

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

case "$1" in
  start)
    echo −n "Starting system log daemon..."
    start−stop−daemon −S −q −o −x /usr/sbin/syslogd −− −m 0
    check_status

    echo −n "Starting kernel log daemon..."
    start−stop−daemon −S −q −o −x /usr/sbin/klogd
    check_status
    ;;

  stop)
    echo −n "Stopping kernel log daemon..."
    start−stop−daemon −K −q −o −p  /var/run/klogd.pid
    check_status

    echo −n "Stopping system log daemon..."
    start−stop−daemon −K −q −o −p /var/run/syslogd.pid
    check_status
    ;;

  reload)
    echo −n "Reloading system load daemon configuration file..."
    start−stop−daemon −K −q −o −s 1 −p /var/run/syslogd.pid
    check_status
    ;;

  restart)
    echo −n "Stopping kernel log daemon..."
    start−stop−daemon −K −q −o −p /var/run/klogd.pid
```

```
    check_status

    echo −n "Stopping system log daemon..."
    start−stop−daemon −K −q −o −p /var/run/syslogd.pid
    check_status

    sleep 1

    echo −n "Starting system log daemon..."
    start−stop−daemon −S −q −o −x /usr/sbin/syslogd −− −m 0
    check_status

    echo −n "Starting kernel log daemon..."
    start−stop−daemon −S −q −o −x /usr/sbin/klogd
    check_status
    ;;

  *)
    echo "Usage: $0 {start|stop|reload|restart}"
    exit 1
    ;;
esac

# End /etc/init.d/sysklogd
```

Setting up symlinks and permissions

Give these files the proper permissions and create the necessary symlinks by running the following commands:

```
root:~#   cd /etc/init.d
root:/etc/init.d#   chmod 755 rcS reboot halt mountfs umountfs
root:/etc/init.d#   chmod 755 sendsignals checkroot sysklogd
root:/etc/init.d#   cd ../rc0.d
root:/etc/rc0.d#   ln -s ../init.d/sysklogd K90sysklogd
root:/etc/rc0.d#   ln -s ../init.d/sendsignals S80sendsignals
root:/etc/rc0.d#   ln -s ../init.d/umountfs S90umountfs
root:/etc/rc0.d#   ln -s ../init.d/halt S99halt
root:/etc/rc0.d#   cd ../rc6.d
root:/etc/rc6.d#   ln -s ../init.d/sysklogd K90sysklogd
root:/etc/rc6.d#   ln -s ../init.d/sendsignals S80sendsignals
root:/etc/rc6.d#   ln -s ../init.d/umountfs S90umountfs
root:/etc/rc6.d#   ln -s ../init.d/reboot S99reboot
root:/etc/rc6.d#   cd ../rcS.d
root:/etc/rcS.d#   ln -s ../init.d/checkroot S05checkroot
root:/etc/rcS.d#   ln -s ../init.d/mountfs S10mountfs
root:/etc/rcS.d#   cd /etc/rc2.d
root:/etc/rc2.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:

```
# Begin /etc/fstab

/dev/<LFS−partition designation> / ext2 defaults 0 1
/dev/<swap−partition designation> none swap sw 0 0
proc /proc proc defaults 0 0

# End /etc/fstab
```

Replace <LFS−partition designation> and <swap−partition designation> with the appropriate devices (/dev/hda5 and /dev/hda6 in my case).

Chapter 7. Setting up basic networking

Installing network software

Installing Netkit−base

Install Netkit−base by running the following commands:

```
root:/usr/src/netkit-base-0.17...#   ./configure --prefix=/usr
root:/usr/src/netkit-base-0.17...#   make
root:/usr/src/netkit-base-0.17...#   make install
root:/usr/src/netkit-base-0.17...#   cd etc.sample
root:/usr/src/netkit-base-0.17.../etc.sample#   cp services \
> protocols /etc
```

Installing Net−tools

Install Net−tools by running the following commands:

```
root:/usr/src/net-tools-1.54#   make
root:/usr/src/net-tools-1.54#   make install
```

Creating network boot scripts

Creating the /etc/init.d/localnet bootscript

Create a new file `/etc/init.d/localnet` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/localnet

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

echo -n "Setting up loopback device..."
/sbin/ifconfig lo 127.0.0.1
check_status

echo -n "Setting up hostname..."
/bin/hostname --file /etc/hostname
check_status

# End /etc/init.d/localnet
```

Setting up permissions and symlink

Set the proper file permissions and create the necessary symlink by running the following commands:

```
root:~#   cd /etc/init.d
root:/etc/init.d#   chmod 755 /etc/init.d/localnet
root:/etc/init.d#   cd ../rcS.d
root:/etc/rcS.d#   ln -s ../init.d/localnet S03localnet
```

Creating the /etc/hostname file

Create a new file `/etc/hostname` and put the hostname in it. This is not the FQDN (Fully Qualified Domain Name). This is the name you wish to call your computer in a network. An example:

```
# Begin /etc/hostname

lfs

# End /etc/hostname
```

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` containing:

```
# Begin /etc/hosts (no network card version)
```

```
127.0.0.1 www.linuxfromscratch.org <contents of /etc/hostname> localhost

# End /etc/hosts (no network card version)
```

If you do configure a network card, create a new file `/etc/hosts` containing:

```
# Begin /etc/hosts (network card version)

127.0.0.1 localhost
192.168.1.1 www.linuxfromscratch.org <contents of /etc/hostname>

# End /etc/hosts (network card version)
```

Of course, change the 192.168.1.1 and www.linuxfromscratch.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 file

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:

```
#!/bin/sh
# Begin /etc/init.d/ethnet

check_status()
{
 if [ $? = 0 ]
 then
  echo "OK"
 else
  echo "FAILED"
 fi
}

IPADDR="209.83.245.12" # Replace with your own IP address
NETMAKSK="255.255.255.0" # Replace with your own Netmask
NETWORK="209.83.245.0" # Replace with your own Network address
BROADCAST="209.83.245.255" # Replace with your own Broadcast addr.
GATEWAY="209.83.245.1" # Replace with your own Gateway address
```

```
echo -n "Setting up eth0..."
/sbin/ifconfig eth0 $(IPADDR) broadcast $(BROADCAST) netmask $(NETMASK)
check_status

echo "Setting up route..."
/sbin/route add -net $(NETWORK) netmask $(NETMASK) eth0
check_status

echo "Adding default gateway..."
/sbin/route add default gw $(GATEWAY) metric 1
check_status

# End /etc/init.d/ethnet
```

Setting up permissions and symlink

Set the proper file permissions and create the necessary symlink by running the following commands:

```
root:~#   cd /etc/init.d
root:/etc/init.d#   chmod 755 /etc/init.d/ethnet
root:/etc/init.d#   cd ../rc2.d
root:/etc/rc2.d#   ln -s ../init.d/ethnet S10ethnet
```

Chapter 8. Making the LFS system bootable

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:

```
root:/usr/src/linux#   make menuconfig
root:/usr/src/linux#   make dep
root:/usr/src/linux#   make bzImage
root:/usr/src/linux#   cp arch/i386/boot/bzImage
/boot/lfskernel
```

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:

```
image=/boot/lfskernel
   label=lfs
   root=<partition>
   read−only
```

<partition> must be replaced by your partition's designation (which would be /dev/hda5 in my case).

Now update the boot loader by running:

```
root:~# lilo
```

Testing the system

Now that all software has been installed, bootscripts have been written and the local network is setup, it's time for you to reboot your computer and test these new scripts to verify that they actually work. You first want to execute them manually from the /etc/init.d directory so you can fix the most obvious problems (typos, wrong paths and such). When those scripts seem to work just fine manually they should also work during a system start or shutdown. There's only one way to test that. 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).

At this point your basic LFS system is ready for use. Everything else that follows now is optional, so you can skip packages at your own discretion. But do keep in mind that if you skip packages (especially libraries) you can break dependencies of other packages. For example, when the Lynx browser is installed, the zlib library is installed as well. You can decide to skip the zlib library, but this library isn't used by Lynx alone. Other packages require this library too. The same may apply to other libraries and programs.

III. Part III – Installation of a basic system on Apple PowerPC systems

Table of Contents

Chapter 9. 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 download a newer version).

Please note that this list used to be ordered on usage, meaning that the first package mentioned in this list was the first package used in this book. That's no longer the case because several chapters have been moved around, so that doens't apply. I didn't have the time to re−order this list in this development release. The next release will have this list ordered again.

- Sysvinit (2.78): ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/
- Bash (2.04): ftp://ftp.gnu.org/gnu/bash
- Linux Kernel (2.2.14): ftp://ftp.kernel.org/pub/linux/kernel/
- Kernel USB patch: 216.22.163.20/usb−2.3.50−1−for−2.2.14.diff.gz
- Binutils (2.9.5.0.37): ftp://ftp.varesearch.com/pub/support/hjl/binutils/
- Bzip2 (0.9.5d): http://sourceware.cygnus.com/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/
- Glibc (2.1.3): ftp://ftp.gnu.org/gnu/glibc/
- Glibc−crypt (2.1.3): ftp://ftp.gwdg.de/pub/linux/glibc/
- Glibc−linuxthreads (2.1.3): ftp://ftp.gnu.org/gnu/glibc/
- Glibc−patch: ftp://216.22.163.20/glibc−2.1.3−ctype.patch
-

- Grep (2.4.2): ftp://ftp.gnu.org/gnu/grep/
- Gzip (1.2.4a): ftp://ftp.gnu.org/gnu/gzip/
- Make (3.78.1): ftp://ftp.gnu.org/gnu/make/
- Ed (0.2): ftp://ftp.gnu.org/gnu/ed/
- Patch (2.5.4): ftp://ftp.gnu.org/gnu/patch/
- Sed (3.02): ftp://ftp.gnu.org/gnu/sed/
- Shell 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/
- Util Linux (2.10h): ftp://ftp.win.tue.nl/pub/linux/utils/util−linux/
- Pmac Utils((1.1.1): ftp://216.22.163.20/pmac−utils−1.1.1−patched.tar.gz
- Bison (1.28): ftp://ftp.gnu.org/gnu/bison/
- Mawk (1.3.3) ftp://ftp.whidbey.net/pub/brennan/
- Find Utils (4.1): ftp://ftp.gnu.org/gnu/findutils/
- Termcap (1.3): ftp://ftp.gnu.org/gnu/termcap/
- Ncurses (4.2): ftp://ftp.gnu.org/gnu/ncurses/
- Less (340): ftp://ftp.gnu.org/gnu/less/
-

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/gnu/flex/
- E2fsprogs (1.18): ftp://tsx−11.mit.edu/pub/linux/packages/ext2fs/
- File (3.26): http://www.linuxfromscratch.org/download/file−3.26−lfs.tar.gz
- Groff (1.15): ftp://ftp.gnu.org/gnu/groff/
- Ld.so (1.9.9): ftp://tsx−11.mit.edu/pub/linux/packages/GCC/
- Libtool (1.3.4): ftp://ftp.gnu.org/gnu/libtool/
- Linux86 (0.14.3): http://www.linuxfromscratch.org/download/linux86−0.14.3−lfs.tar.gz
- Shadow Password Suite (19990827): ftp://piast.t19.pwr.wroc.pl/pub/linux/shadow/
- Man (1.5h1): ftp://ftp.win.tue.nl/pub/linux−local/utils/man/
- Modutils (2.3.9): ftp://ftp.ocs.com.au/pub/modutils/
- Procinfo (17): ftp://ftp.cistron.nl/pub/people/svm/
- Procps (2.0.6): ftp://people.redhat.com/johnsonm/procps/
-

Psmisc (19): ftp://lrcftp.epfl.ch/pub/linux/local/psmisc/

- Start−stop−daemon (0.4.1): http://www.linuxfromscratch.org/download/ssd−0.4.1.tar.gz
- Sysklogd (1.3.31): ftp://sunsite.unc.edu/pub/Linux/system/daemons/
- Vim−rt + Vim−src (5.6): ftp://ftp.vim.org/pub/editors/vim/unix/

Chapter 10. Preparing a new partition

How we are going to do things

We are going to build the LFS system 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 safe yourself a lot of trouble if your normal system uses glibc−2.0 or newer. Libc5 isn't supported by this book, though it isn't impossible to use a libc5 system if you have no choice.

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 at least 5 00 MB. You can get away with around 250MB for a bare system with no extra bells and whistles (such as software for emailing, networking, Internet, X Window System and such). If you already have a Linux Native partition available, you can skip this subsection.

Start the pdisk program (or some other fdisk program you prefer) with the appropriate hard disk as the option (like /dev/shda if you want to create a new partition on the first SCSI disk). The partition that is available for partitioning is called *Apple_Free_Space*. To create a linux capable partition in that free space, type c followed by the partition designation of the free space p<n>, the size in MB of the desired partition <size>M and the name of the partition <name>. The example below creates a 1.8 GB partition name root starting at the beginning of the free space designated as partition 6: *c p6 1800M root*

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 was sda5, you would run:

```
root:~# mke2fs /dev/sda5
```

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 date to it) is mounting it. If you mount it under /mnt/sda5, you can access this partition by going to the /mnt/sda5 directory and then do whatever you need to do. This document will assume that you have mounted the partition on a subdirectory under /mnt. It doesn't matter which directory you choose (or you can use just the /mnt directory as the mount point), but a good practice is to create a directory with the same name as the partition's designation. In my case the LFS partition is called sda5 and therefore I mount it on /mnt/sda5

Create the /mnt directory if it doesn't exist yet by runnning:

```
root:~# mkdir /mnt
```

Create the /mnt/xxx directory by running:

```
root:~# mkdir /mnt/xxx
```

Replace "xxx" by your partition's designation.

Now mount the LFS partition by running:

```
root:~# mount /dev/xxx  /mnt/xxx
```

Replace "xxx" by your partition's designation.

This directory (/mnt/xxx) 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/xxx/etc" where "xxx" is replaced by your partition's designation.

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:

```
root:~#   cd $LFS
root:/mnt/sda5#   mkdir bin boot dev etc home lib mnt proc
root \
> sbin tmp usr var
root:/mnt/sda5#   cd $LFS/usr
root:/mnt/sda5/usr#   mkdir bin include lib local sbin share
src
root:/mnt/sda5/usr#   ln -s share/man man
root:/mnt/sda5/usr#   ln -s share/doc doc
root:/mnt/sda5/usr#   ln -s share/info info
root:/mnt/sda5/usr#   ln -s ../etc etc
root:/mnt/sda5/usr#   ln -s ../var var
root:/mnt/sda5/usr#   cd $LFS/usr/share
root:/mnt/sda5/usr/share#   mkdir dict doc info locale man
nls \
> misc terminfo zoneinfo
root:/mnt/sda5/usr/share#   cd $LFS/usr/share/man
root:/mnt/sda5/usr/share/man#   mkdir man1 man2 man3 man4
man5 \
> man6 man7 man8
root:/mnt/sda5/usr/share/man#   cd $LFS/var
root:/mnt/sda5/var#   mkdir lock log run spool 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).

Copying the /dev directory

We can create every single file that we need to be in the $LFS/dev directory using the mknod command, but that just takes up a lot of time. I choose to just simply copy the current /dev directory to the $LFS partition. Use this command to copy the entire directory while preserving original rights, symlinks and ownerships:

```
root:~# cp −av /dev $LFS
root:~#   chown root $LFS/dev/*
```

I'm aware that this isn't the best way to create the files. I know of a MAKEDEV script but I choose not to use it. I'm actually waiting for the 2.4 Linux kernel to be released. The kernel has a stable version of the devfs which this book will use in the future. Devfs is a dynamic file system which makes the static files in /dev obsolete. You mount the dev file system to a mount point (kind of like the way the proc file system works) and the kernel will create the files in /dev you need on−the−fly. So the waiting is for the next stable kernel to be released.

Chapter 11. Installing basic system software

In this chapter 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 optional issues such as setting up networking, Internet servers + clients (telnet, ftp, http, email), setting up Internet itself and the X Window System. You can skip chapters at your own discretion. If you don't plan on going online with the LFS system there's little use to setup Internet for example.

This chapter is devided in two chunks. The first part installs a few necessary programs on the LFS system. These programs are needed to install the rest of the programs that belong to a basic system. When the first part is done, we will enter a chroot'ed environment. This means that we start a shell with $LFS as the root directory (instead of the usual / directory as the root directory). This has the same effect as rebooting the computer into the LFS system, but this way we don't have to reboot. If something goes wrong, you don't need to reboot back in the normal Linux system to fix whatever you need to fix. You just open a new shell on a virtual console, or start a new xterm and you can do what you need to do.

The software in the first part will be linked statically. These programs will be re−installed in the second part and linked dynamically. The reason for the static version first is that there is a chance that our normal Linux system and our LFS system−to−be don't use the same C Library versions. If the programs in the first part are linked against an older C library version, those program might not work too well on the LFS system.

About debugging symbols

Every program and library is by default compiled with debugging symbols. This means you can run a program or library through a debugger and the debugger's output will be more user friendly. These debugging symbols also enlarge the program or library significantly. This document will not install software without debugging symbols (as I don't know if the majority of readers do or do not debug software). In stead, you can remove those symbols manually if you want with the strip program.

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/*).

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

- A static Bash binary with debugging symbols: 2.3MB
- A static Bash binary without debugging symbols: 645KB
- A dynamic Bash binary with debugging symbols: 1.2MB
- A dynamic Bash binary without debugging symbols: 478KB
- $LFS/lib and $LFS/usr/lib (glibc and gcc files) with debugging symbols: 87MB
- $LFS/lib and $LFS/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. The difference would be even greater when I would do this at the end of this book when everything is installed.

Preparing the LFS system for installing basic system software

Installing Bash

Install Bash by running the following commands:

```
root:/mnt/sda5/usr/src/bash-2.04#   ./configure
--enable-static-link
root:/mnt/sda5/usr/src/bash-2.04#   make
root:/mnt/sda5/usr/src/bash-2.04#   make -e prefix=$LFS/usr
install
root:/mnt/sda5/usr/src/bash-2.04#   mv $LFS/usr/bin/bash
$LFS/bin
root:/mnt/sda5/usr/src/bash-2.04#   cd $LFS/bin
root:/mnt/sda5/bin#   ln -s bash sh
```

Installing Binutils

Install Binutils by running the following commands:

```
root:/mnt/sda5/usr/src/binutils-2.9.5.0.37#   ./configure \
> --prefix=/usr
root:/mnt/sda5/usr/src/binutils-2.9.5.0.37#   make -e \
> LDFLAGS=-all-static
root:/mnt/sda5/usr/src/binutils-2.9.5.0.37a#   make -e \
> prefix=$LFS/usr install
```

Installing Bzip2

Before we can install Bzip2 we need to modify the `Makefile` file. Open the `Makefile` file in a text editor and find the lines that start with *$(CC) $(CFLAGS) -o*

Replace those parts with: *$(CC) $(CFLAGS) $(LDFLAGS) -o*

Now install Bzip2 by running the following commands:

```
root:/mnt/sda5/usr/src/bzip2-0.9.5d#   make -e LDFLAGS=-static
root:/mnt/sda5/usr/src/bzip2-0.9.5d#   make -e
PREFIX=$LFS/usr \
> install
root:/mnt/sda5/usr/src/bzip2-0.9.5d#   cd $LFS/usr/bin
root:/mnt/sda5/usr/bin#   mv bunzip2 bzip2 $LFS/bin
```

Installing Diffutils

Install Diffutils by running the following commands:

```
root:/mnt/sda5/usr/src/diffutils-2.7#   ./configure
--prefix=/usr
root:/mnt/sda5/usr/src/diffutils-2.7#   make -e
LDFLAGS=-static
root:/mnt/sda5/usr/src/diffutils-2.7#   make -e
prefix=$LFS/usr \
> install
```

This package is known to cause static link problems on certain platforms. If you're having trouble compiling this package as well, you can download a patch from http://www.linuxfromscratch.org/download/diffutils–2.7.patch.gz

Install this patch by running the following command:

```
root:/mnt/sda5/usr/src/diffutils-2.7#    patch -Np1 \
> -i ../diffutils-2.7.patch
```

Now recompile the package using the same commands as above.

Installing Fileutils

Install Fileutils by running the following commands:

```
root:/mnt/sda5/usr/src/fileutils-4.0#   ./configure \
> --disable-nls --prefix=/usr
root:/mnt/sda5/usr/src/fileutils-4.0#   make -e
LDFLAGS=-static
root:/mnt/sda5/usr/src/fileutils-4.0#   make -e
prefix=$LFS/usr \
> install
root:/mnt/sda5/usr/src/fileutils-4.0#   cd $LFS/usr/bin
root:/mnt/sda5/usr/bin#   mv chgrp chmod chown cp dd df ln
$LFS/bin
root:/mnt/sda5/usr/bin#   mv ls mkdir mknod mv rm rmdir sync
$LFS/bin
```

Installing 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. Many glibc−2.0 based systems have gcc−2.7.2.3 installed and you can't compile glibc−2.1.3 with that compiler. Many glibc−2.1 based systems have egcs−2.95.x installed and that version doesn't work too well either (sometimes it works fine, sometimes it doesn't depending on various circumstances).

If you normal Linux system does not have gcc−2.95.2 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.

Install GCC by running the following commands:

```
root:/mnt/sda5/usr/src#   mkdir $LFS/usr/src/gcc-build
root:/mnt/sda5/usr/src#   cd $LFS/usr/src/gcc-build
root:/mnt/sda5/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++
root:/mnt/sda5/usr/src/gcc-build#   make bootstrap
root:/mnt/sda5/usr/src/gcc-build#   make install
```

Installing GCC on the LFS system

Install GCC by running the following commands:

```
root:/mnt/sda5/usr/src#   mkdir $LFS/usr/src/gcc-build
root:/mnt/sda5/usr/src#   cd $LFS/usr/src/gcc-build
root:/mnt/sda5/usr/src/gcc-build#   ../gcc-2.95.2/configure \
> --prefix=/usr --with-local-prefix=/usr \
> --with-gxx-include-dir=/usr/include/g++ \
> --enable-languages=c,c++ --disable-nls
root:/mnt/sda5/usr/src/gcc-build#   make -e LDFLAGS=-static
bootstrap
root:/mnt/sda5/usr/src/gcc-build#   make -e prefix=$LFS/usr \
> local_prefix=$LFS/usr gxx_include_dir=$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:

```
root:~#   cd $LFS/lib
root:/mnt/sda5/lib#   ln -s
../usr/lib/gcc-lib/<host>/2.95.2/cpp cpp
root:/mnt/sda5/lib#   cd $LFS/usr/lib
root:/mnt/sda5/usr/lib#   ln -s gcc-lib/<host>/2.95.2/cpp cpp
root:/mnt/sda5/usr/lib#   cd $LFS/usr/bin
root:/mnt/sda5/usr/bin#   ln -s gcc cc
```

Replace <host> with the directory where the gcc−2.95.2 files are installed (which is i686−unknown−linux in my case).

Installing Glibc

A note on the glibc−crypt package

An excerpt from the README file that is distributed with the glibc−crypt package:

The add−on is not included in the main distribution of the GNU C library because some governments, most notably those of France, Russia, and the US, have very restrictive rules governing the distribution and use of encryption software. Please read the node "Legal Problems" in the manual for more details.

In particular, the US does not allow export of this software without a licence, including via the Internet. So please do not download it from the main FSF FTP site at ftp.gnu.org if you are outside the US. This software was completely developed outside the US.

"This software" refers to the glibc−crypt package at ftp://ftp.gwdg.de/pub/linux/glibc/. This law only affects people who don't live in the US. It's not prohibited to import DES software, so if you live in the US you can import the file safely from Germany without breaking cryptographic laws. This law is changing lately and I don't know what the status of it is at the moment. Better be safe than sory.

Installing Glibc

Copy the Glibc−crypt and Glibc−linuxthreads archives into the unpacked glibc directory. Copy the glibc−2.1.3−ctype.patch file to $LFS/usr/src

Unpack the glibc−crypt and glibc−linuxthreads archives there, but don't enter the created directories. Just unpack 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 in that directory you create a new file `configparms` containing:

```
# Begin configparms

slibdir=/lib
sysconfdir=/etc

# End configparms
```

Change to the `$LFS/usr/src/glibc-2.1.3` directory and install Glibc by running the following commands if your system already had a suitable GCC version installed:

```
root:/mnt/sda5/usr/src/glibc-2.1.3#   patch -p1 <
../glibc-2.1.3-ctype.patch
root:/mnt/sda5/usr/src/glibc-2.1.3#   cd ../glibc-build
root:/mnt/sda5/usr/src/glibc-build#
../glibc-2.1.3/configure \
> --prefix=/usr --enable-add-ons
root:/mnt/sda5/usr/src/glibc-build#   make
root:/mnt/sda5/usr/src/glibc-build#   make install_root=$LFS
install
```

Change to the $LFS/usr/src/glibc-build directory and install Glibc by running the following command if your system did not already have a suitable GCC version installed and you just installed GCC−2.95.2 on your normal Linux system a little while ago:

```
root:/mnt/sda5/usr/src/glibc-build#
CC=/usr/local/gcc2952/bin/gcc \
> ../glibc-2.1.3/configure --prefix=/usr \
> --enable-add-ons
root:/mnt/sda5/usr/src/glibc-build#  make
root:/mnt/sda5/usr/src/glibc-build#  make install_root=$LFS
install
```

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,userid's and groupid's. You can check which C library version your normal Linux system uses by running:

```
root:~# ls /lib/libc*
```

Your system uses glib−2.0 if there is a file that looks like *libc−2.0.7.so*

Your system uses glibc−2.1 if there is a file that looks like *libc−2.1.3.so*

Of course, the micro version number can be different (you could have libc−2.1.2 or libc−2.1.1 for example).

If you have a libc−2.0.x file copy the NSS library files by running:

```
root:~# cp -av /lib/libnss*  $LFS/lib
```

There are a few distributions that don't have files from which you can see which version of the C Library it is. If that's the case, it will be hard to determine which C library version you exactly have. Try to obtain this information using your distribution's installation tool. It often says which version it has available. If you can't figure out at all which C Library version is used, then copy the NSS files anyway and hope for the best. That's the best advise I can give I'm afraid.

Installing Grep

Install Grep by running the following commands:

```
root:/mnt/sda5/usr/src/grep-2.4.2#   ./configure
--prefix=/usr \
> --disable-nls
root:/mnt/sda5/usr/src/grep-2.4.2#   make -e LDFLAGS=-static
root:/mnt/sda5/usr/src/grep-2.4.2#   make -e prefix=$LFS/usr
install
```

This package is known to cause static linking problems on certain platforms. If you're having trouble compiling this package as well, you can download a patch from http://www.linuxfromscratch.org/download/grep–2.4.2.patch.gz

Install this patch by running the following command:

```
root:/mnt/hda5/usr/src/grep-2.4.2#    patch -Np1 \
> -i ../grep-2.4.2.patch
```

Now recompile the package using the same commands as above.

Installing Gzip

Install Gzip by running the following commands:

```
root:/mnt/sda5/usr/src/gzip-1.2.4a#   ./configure
--prefix=/usr
root:/mnt/sda5/usr/src/gzip-1.2.4a#   make -e LDFLAGS=-static
root:/mnt/sda5/usr/src/gzip-1.2.4a#   make -e prefix=$LFS/usr
install
root:/mnt/sda5/usr/src/gzip-1.2.4a#   cd $LFS/usr/bin
root:/mnt/sda5/usr/bin#   mv gunzip gzip $LFS/bin
```

This package is known to cause compilation problems on certain platforms. If you're having trouble compiling this package as well, you can download a fixed package from http://www.linuxfromscratch.org/download/gzip−1.2.4a.patch.gz

Install this patch by running the following command:

```
root:/usr/src/#    patch −Np1 \
> −i ../findutils−4.1.patch.gz
```

Now recompile the package using the same commands as above.

Installing Make

Install Make by running the following commands:

```
root:/mnt/sda5/usr/src/make−3.78.1#   ./configure
−−prefix=/usr
root:/mnt/sda5/usr/src/make−3.78.1#   make −e LDFLAGS=−static
root:/mnt/sda5/usr/src/make−3.78.1#   make −e prefix=$LFS/usr
install
```

Installing Sed

Install Sed by running the following commands:

```
root:/mnt/sda5/usr/src/sed−3.02#   ./configure −−prefix=/usr
root:/mnt/sda5/usr/src/sed−3.02#   make −e LDFLAGS=−static
root:/mnt/sda5/usr/src/sed−3.02#   make −e prefix=$LFS/usr
install
root:/mnt/sda5/usr/src/sed−3.02#   mv $LFS/usr/bin/sed
$LFS/bin
```

This package is known to cause static linking problems on certain platforms. If you're having trouble compiling this package as well, you can download a patch from http://www.linuxfromscratch.org/download/sed−3.02.patch.gz

Install this patch by running the following command:

```
root:/usr/src/#    patch -Np1 \
> -i ../sed-3.02.patch.gz
```

Now recompile the package using the same commands as above.

Installing Shellutils

Install Shellutils by running the following commands:

```
root:/mnt/sda5/usr/src/sh-utils-2.0#   ./configure
--prefix=/usr \
> --disable-nls
root:/mnt/sda5/usr/src/sh-utils-2.0#   make -e LDFLAGS=-static
root:/mnt/sda5/usr/src/sh-utils-2.0#   make -e
prefix=$LFS/usr \
> install
root:/mnt/sda5/usr/src/sh-utils-2.0#   cd $LFS/usr/bin
root:/mnt/sda5/usr/bin#   mv date echo false pwd stty $LFS/bin
root:/mnt/sda5/usr/bin#   mv su true uname hostname $LFS/bin
```

Installing Tar

Install Tar by running the following commands:

```
root:/mnt/sda5/usr/src/tar-1.13#   ./configure --prefix=/usr \
> --disable-nls
root:/mnt/sda5/usr/src/tar-1.13#   make -e LDFLAGS=-static
root:/mnt/sda5/usr/src/tar-1.13#   make -e prefix=$LFS/usr
install
root:/mnt/sda5/usr/src/tar-1.13#   mv $LFS/usr/bin/tar
$LFS/bin
```

Installing Textutils

Install Textutils by running the following commands:

```
root:/mnt/sda5/usr/src/textutils-2.0#   ./configure
--prefix=/usr \
> --disable-nls
root:/mnt/sda5/usr/src/textutils-2.0#   make -e
LDFLAGS=-static
root:/mnt/sda5/usr/src/textutils-2.0#   make -e
prefix=$LFS/usr \
> install
root:/mnt/sda5/usr/src/textutils-2.0#   mv $LFS/usr/bin/cat
$LFS/bin
```

Creating passwd and group files

Create a new file `$LFS/etc/passwd` containing the following:

```
# Begin /etc/passwd
root:kRKb4s/LH8yDQ:0:0:root:/root:/bin/bash
# End /etc/passwd
```

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

```
# Begin /etc/group
root::0:
# End /etc/group
```

The encoded password above is: *lfs123*

Installing basic system software

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.

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

Entering the chroot'ed environment

Enter the following commands to setup 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/xxx but the shell doesn't know that).

```
root:~#   cd $LFS
root:/mnt/sda5#   chroot $LFS bash
```

Installing 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 setup the proper symlinks in /usr/include. Create the `/usr/include/linux` and `/usr/include/asm` symlinks by running the following commands:

```
root:~#   cd /usr/include
root:/usr/include#   ln -s ../src/linux/include/linux linux
root:/usr/include#   ln -s ../src/linux/include/asm-ppc asm
```

Installing Ed

Install Ed by running the following commands:

```
root:/usr/src/ed-0.2#    ./configure --prefix=/usr
root:/usr/src/ed-0.2#    make
root:/usr/src/ed-0.2#    make install
```

Installing Patch

Install Patch by running the following commands:

```
root:/usr/src/patch-2.5.4#   ./configure --prefix=/usr
root:/usr/src/patch-2.5.4#   make
root:/usr/src/patch-2.5.4#   make install
```

Installing GCC

Install GCC by running the following commands:

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

Installing Bison

Install Bison by running the following commands:

```
root:/usr/src/bison-1.28#   ./configure --prefix=/usr \
> --datadir=/usr/share/bison
root:/usr/src/bison-1.28#   make
root:/usr/src/bison-1.28#   make install
```

Installing Mawk

Install Mawk by running the following commands:

```
root:/usr/src/mawk-1.3.3#   ./configure
root:/usr/src/mawk-1.3.3#   make
root:/usr/src/mawk-1.3.3#   make -e BINDIR=/usr/bin \
> MANDIR=/usr/share/man/man1 install
root:/usr/src/mawk-1.3.3#   cd /usr/bin
root:/usr/bin#   ln -s mawk awk
```

Installing Findutils

Install Findutils by running the following commands:

```
root:/usr/src/findutils-4.1#   ./configure --prefix=/usr
root:/usr/src/findutils-4.1#   make
root:/usr/src/findutils-4.1#   make install
```

This package is known to cause compilation problem. If you're having trouble compiling this package as well, you can download a patch from http://www.linuxfromscratch.org/download/findutils–4.1.patch.gz

Install this patch by running the following command:

```
root:/usr/src/#    patch -Np1 \
> -i ../findutils-4.1.patch.gz
```

Now recompile the package using the same commands as above.

Installing Termcap

Install Termcap by running the following commands:

```
root:/usr/src/termcap-1.3#   ./configure --prefix=/usr
root:/usr/src/termcap-1.3#   make
root:/usr/src/termcap-1.3#   make install
```

Installing Ncurses

Install Ncurses by running the following commands:

```
root:/usr/src/ncurses-4.2#   ./configure --prefix=/usr
--with-shared
root:/usr/src/ncurses-4.2#   make
root:/usr/src/ncurses-4.2#   make install
```

Installing Less

Install Less by running the following commands:

```
root:/usr/src/less-340#   ./configure --prefix=/usr
root:/usr/src/less-340#   make
root:/usr/src/less-340#   make install
root:/usr/src/less-340#   mv /usr/bin/less /bin
```

Installing Perl

Install Perl by running the following commands:

```
root:/usr/src/perl-5.6.0#   ./Configure
root:/usr/src/perl-5.6.0#   make
root:/usr/src/perl-5.6.0#   make test
root:/usr/src/perl-5.6.0#   make install
```

Note that you have to change the installation path to `/usr` yourself. The Perl installation defaults to the /usr/local/subdir

Also note that a few tests during the make test phase will fail because we don't have network support installed yet.

Installing M4

Install M4 by running the following commands:

```
root:/usr/src/m4-1.4#   ./configure --prefix=/usr
root:/usr/src/m4-1.4#   make
root:/usr/src/m4-1.4#   make install
```

Installing Texinfo

Install Texinfo by running the following commands:

```
root:/usr/src/texinfo-4.0#   ./configure --prefix=/usr
root:/usr/src/texinfo-4.0#   make
root:/usr/src/texinfo-4.0#   make install
```

Installing Autoconf

Install Autoconf by running the following commands:

```
root:/usr/src/autoconf-2.13#   ./configure --prefix=/usr
root:/usr/src/autoconf-2.13#   make
root:/usr/src/autoconf-2.13#   make install
```

Installing Automake

Install Automake by running the following commands:

```
root:/usr/src/automake-1.4#   ./configure --prefix=/usr
root:/usr/src/automake-1.4#   make install
```

Installing Bash

Install Bash by running the following commands:

```
root:/usr/src/bash-2.04#   ./configure --prefix=/usr
root:/usr/src/bash-2.04#   make
root:/usr/src/bash-2.04#   make install
root:/usr/src/bash-2.04#   mv /usr/bin/bash /bin
```

Installing Flex

Install Flex by running the following commands:

```
root:/usr/src/flex-2.5.4a#   ./configure --prefix=/usr
root:/usr/src/flex-2.5.4a#   make
root:/usr/src/flex-2.5.4a#   make install
```

Installing Binutils

Install Binutils by running the following commands:

```
root:/usr/src/binutils-2.9.5.0.37#   ./configure --prefix=/usr
root:/usr/src/binutils-2.9.5.0.37#   make
```

```
root:/usr/src/binutils-2.9.5.0.37#   make install
```

Installing Bzip2

Install Bzip2 by running the following commands:

```
root:/usr/src/bzip2-0.9.5d#   make
root:/usr/src/bzip2-0.9.5d#   make PREFIX=/usr install
root:/usr/src/bzip2-0.9.5d#   cd /usr/bin
root:/usr/bin#   mv bunzip2 bzip2 /bin
```

Installing Diffuitls

Install Diffutils by running the following commands:

```
root:/usr/src/diffutils-2.7#   ./configure --prefix=/usr
root:/usr/src/diffutils-2.7#   make
root:/usr/src/diffutils-2.7#   make install
```

Installing E2fsprogs

Install E2fsprogs by running the following commands:

```
root:/usr/src/e2fsprogs-1.18#   ./configure --prefix=/usr
root:/usr/src/e2fsprogs-1.18#   make
root:/usr/src/e2fsprogs-1.18#   make install
root:/usr/src/e2fsprogs-1.18#   cd /usr/sbin
root:/usr/sbin#   mv *e2* *fs* mklost+found /sbin
```

Installing File

Install File by running the following commands:

```
root:/usr/src/file-3.26#   ./configure --prefix=/usr
root:/usr/src/file-3.26#   make
root:/usr/src/file-3.26#   make install
```

Installing Fileutils

Install Fileutils by running the following commands:

```
root:/usr/src/fileutils-4.0#   ./configure --prefix=/usr
root:/usr/src/fileutils-4.0#   make
root:/usr/src/fileutils-4.0#   make install
root:/usr/src/fileutils-4.0#   cd /usr/bin
root:/usr/bin#   mv chgrp chmod chown cp dd df ln /bin
root:/usr/bin#   mv ls mkdir mknod mv rm rmdir sync /bin
```

Installing Grep

Install Grep by running the following commands:

```
root:/usr/src/grep-2.4.2#   ./configure --prefix=/usr
root:/usr/src/grep-2.4.2#   make
root:/usr/src/grep-2.4.2#   make install
```

Installing Groff

Install Groff by running the following commands:

```
root:/usr/src/groff-1.15#   ./configure --prefix=/usr
root:/usr/src/groff-1.15#   make
root:/usr/src/groff-1.15#   make install
```

Installing Gzip

Install Gzip by running the following commands:

```
root:/usr/src/gzip-1.2.4a#   ./configure --prefix=/usr
root:/usr/src/gzip-1.2.4a#   make
root:/usr/src/gzip-1.2.4a#   make install
root:/usr/src/gzip-1.2.4a#   cd /usr/bin
root:/usr/bin#   mv gunzip gzip /bin
```

Installing Ld.so

Install Ld.so by running the following commands:

```
root:/usr/src/ld.so-1.9.10#   cd util
root:/usr/src/ld.so-1.9.10/util#   make ldd ldconfig
root:/usr/src/ld.so-1.9.10/util#   cp ldd /bin
root:/usr/src/ld.so-1.9.10/util#   cp ldconfig /sbin
root:/usr/src/ld.so-1.9.10/util#   rm /usr/bin/ldd
```

Installing Libtool

Install Libtool by running the following commands:

```
root:/usr/src/libtool-1.3.4#   ./configure --prefix=/usr
root:/usr/src/libtool-1.3.4#   make
root:/usr/src/libtool-1.3.4#   make install
```

Installing Linux86

Install Linux86 by running the following commands:

```
root:/usr/src/linux-86#   cd as
root:/usr/src/linux-86/as#   make
root:/usr/src/linux-86#   make install
root:/usr/src/linux-86#   cd ../ld
root:/usr/src/linux-86#   make ld86
root:/usr/src/linux-86#   make install
```

Installing Make

Install Make by running the following commands:

```
root:/usr/src/make-3.78.1#   ./configure --prefix=/usr
root:/usr/src/make-3.78.1#   make
root:/usr/src/make-3.78.1#   make install
```

Installing Shell Utils

Install Shellutils by running the following commands:

```
root:/usr/src/sh-utils-2.0#   ./configure --prefix=/usr
root:/usr/src/sh-utils-2.0#   make
root:/usr/src/sh-utils-2.0#   make install
root:/usr/src/sh-utils-2.0#   cd /usr/bin
root:/usr/bin#   mv date echo false pwd stty /bin
root:/usr/bin#   mv su true uname hostname /bin
```

Installing Shadow Password Suite

Install the Shadow Password Suite by running the following commands:

```
root:/usr/src/shadow-19990827#   ./configure --prefix=/usr
root:/usr/src/shadow-19990827#   make
root:/usr/src/shadow-19990827#   make install
root:/usr/src/shadow-19990827#   cd etc
root:/usr/src/shadow-19990827/etc#   cp limits login.access \
> login.defs.linux shells suauth /etc
root:/usr/src/shadow-19990827#   mv /etc/login.defs.linux \
> /etc/login.defs
```

Installing Man

Install Man by running the following commands:

```
root:/usr/src/man-1.5h1#   ./configure -default
root:/usr/src/man-1.5h1#   make
root:/usr/src/man-1.5h1#   make install
```

Installing Modutils

Install Modutils by running the following commands:

```
root:/usr/src/modutils-2.3.9#   ./configure
root:/usr/src/modutils-2.3.9#   make
root:/usr/src/modutils-2.3.9#   make install
```

Installing Procinfo

Install Procinfo by running the following commands:

```
root:/usr/src/procinfo-17#   make
root:/usr/src/procinfo-17#   make install
```

Installing Procps

Install Procps by running the following commands:

```
root:/usr/src/procps-2.0.6#   gcc -O3 -Wall -Wno-unused -c
watch.c
root:/usr/src/procps-2.0.6#   make
root:/usr/src/procps-2.0.6#   make -e XSCPT="" install
root:/usr/src/procinfo-17#   mv /usr/bin/kill /bin
```

Installing Psmisc

Install Psmisc by running the following commands:

```
root:/usr/src/psmisc-19#   make
root:/usr/src/psmisc-19#   make install
```

Installing Sed

Install Sed by running the following commands:

```
root:/usr/src/sec-3.02#   ./configure --prefix=/usr
root:/usr/src/sec-3.02#   make
root:/usr/src/sec-3.02#   make install
root:/usr/src/sec-3.02#   mv /usr/bin/sed /bin
```

Installing Start−stop−daemon

Install Start−stop−daemon by running the following commands:

```
root:/usr/src/ssd-0.4.1#   make
root:/usr/src/ssd-0.4.1#   make install
```

Installing Sysklogd

Install Sysklogd by running the following commands:

```
root:/usr/src/sysklogd-1.3-31#   make
root:/usr/src/sysklogd-1.3-31#   make install
```

Installing Sysvinit

Install Sysvinit by running the following commands:

```
root:/usr/src/sysvinit-2.78#   cd src
root:/usr/src/sysvinit-2.78#   make
root:/usr/src/sysvinit-2.78#   make install
```

Installing Tar

Install Tar by running the following commands:

```
root:/usr/src/tar-1.13#   ./configure --prefix=/usr
root:/usr/src/tar-1.13#   make
root:/usr/src/tar-1.13#   make install
root:/usr/src/tar-1.13#   mv /usr/bin/tar /bin
```

Installing Textutils

Install Textutuils by running the following commands:

```
root:/usr/src/textutils-2.0#   ./configure --prefix=/usr
root:/usr/src/textutils-2.0#   make
root:/usr/src/textutils-2.0#   make install
root:/usr/src/textutils-2.0#   mv /usr/bin/cat /bin
```

Installing Vim

You need to unpack both the vim−rt and vim−src packages to install Vim. Install Vim by running the following commands:

```
root:/usr/src/vim-5.6#   ./configure --prefix=/usr
root:/usr/src/vim-5.6#   make
root:/usr/src/vim-5.6#   make install
root:/usr/src/vim-5.6#   cd /usr/bin
root:/usr/bin#   ln -s vim vi
```

Installing Util−Linux

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

HAVE_PASSWD=yes
HAVE_SLN=yes
HAVE_TSORT=yes

Install Util−Linux by running the following commands:

```
root:/usr/src/util-linux-2.10h#   groupadd -g 5 tty
root:/usr/src/util-linux-2.10h#   ./configure
root:/usr/src/util-linux-2.10h#   make
root:/usr/src/util-linux-2.10h#   make install
```

Installing Pmac−utils

Install Pmac−utils by running the following commands:

```
root:/usr/src/pmac-utils-1.1.1#   make clock
root:/usr/src/pmac-utils-1.1.1#   cp clock /sbin
root:/usr/src/pmac-utils-1.1.1#   rm /sbin/hwclock
```

Create a new file `/sbin/hwclock` containing the following:

```
#!/bin/sh
# Begin /sbin/hwclock

/sbin/clock -s

# End /sbin/hwclock
```

Set the right permissions by running the following command:

```
chmod 755 /sbin/hwclock
```

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:

```
root:~#    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 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 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` containing:

```
# Begin /etc/nsswitch.conf

passwd: files
group: files
shadow: 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
```

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:

```
root:~#   cd /etc
root:/etc#   rm localtime
root:/etc#   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 Sysklogd

Create the `/etc/syslog.conf` file containing the following:

```
# Begin /etc/syslog.conf

auth,authpriv.* −/var/log/auth.log
*.*;auth,authpriv.none −/var/log/sys.log
daemon.* −/var/log/daemon.log
kern.* −/var/log/kern.log
mail.* −/var/log/mail.log
user.* −/var/log/user.log
*.emerg *

# End /etc/syslog.conf
```

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. 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 shadowed passwords.

If you decide you don't want to use shadowed passwords (after you're read the doc/HOWTO document), you still use this archive since the utilities in this archive are also used on system which have shadowed passwords disabled. You can read all about this in the HOWTO. Also note that you can switch between shadow and non−shadow at any point you want.

Now is a very good moment to read chapter 5 of the doc/HOWTO file. You can read how you can test if shadowing works and if not, how to disable it. If it doesn't work and you haven't tested it, you'll end up with an unusable system after you logout of all your consoles, since you won't be able to login anymore. You can easily fix this by passing the init=/sbin/sulogin parameter to the kernel, unpack the util−linux archive, go to the login−utils directory, build the login program and replace the /bin/login by the one in the util−linux package. Things are never hopelessly messed up (at least not under Linux), but you can avoid a hassle by testing properly and reading manuals ;)

Configuring Sysvinit

Create a new file `/etc/inittab` containing the following:

```
# Begin /etc/inittab

id:2:initdefault:

si::sysinit:/etc/init.d/rcS

su:S:wait:/sbin/sulogin

l0:0:wait:/etc/init.d/rc 0
l1:1: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

ft:6:respawn:/sbin/sulogin

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

1:2345:respawn:/sbin/agetty /dev/tty1 9600
2:2345:respawn:/sbin/agetty /dev/tty2 9600
3:2345:respawn:/sbin/agetty /dev/tty3 9600
4:2345:respawn:/sbin/agetty /dev/tty4 9600
5:2345:respawn:/sbin/agetty /dev/tty5 9600
6:2345:respawn:/sbin/agetty /dev/tty6 9600

# End /etc/inittab
```

Creating the /var/run/utmp file

Programs like login, shutdown, uptime and others want to read from and write to the /var/run/utmp file. This file contains information about who is currently logged in. It also contains information on when the computer was last booted and shutdown.

Create the `/var/run/utmp` and give it the proper permissions by running the following commands:

```
root:~# touch /var/run/utmp
root:~# chmod 644 /var/run/utmp
```

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` containing the following:

```
set nocompatible
set bs=2
```

Chapter 12. Creating system boot scripts

These bootscripts are started at system boot time. The scripts are responsible for mounting the root file system in read−write mode, activating swap, setting up some system settings and starting the various daemons that our system needs.

Preparing the directories and master files

You need the Sysvinit package again for this section.

We need to start by creating a few extra directories that are used by the boot scripts. Create these directories by running:

```
root:~#   cd /etc
root:/etc#   mkdir rc0.d rc1.d rc2.d rc3.d
root:/etc#   mkdir rc4.d rc5.d rc6.d init.d rcS.d
```

Go to the unpacked Sysvinit source directory and copy the `Debian/etc/init.d/rc` file to `/etc/init.d` by running:

```
root:~#   cp /usr/src/sysvinit-2.78/debian/etc/init.d/rc
/etc/init.d
```

The second boot script is called `rcS` and we'll create that one from scratch. Create a new file `/etc/init.d/rcS` containing the following:

```
#!/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
```

Creating the reboot script

Create a new file `/etc/init.d/reboot` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/reboot

echo "System reboot in progress..."

/sbin/reboot -d -f -i

# End /etc/init.d/reboot
```

Creating the halt script

Create a new file `/etc/init.d/halt` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/halt

/sbin/halt –d –f –i –p

# End /etc/init.d/halt
```

Creating the mountfs script

Create a new file `/etc/init.d/mountfs` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/mountfs

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

echo -n "Remounting root file system in read-write mode..."
/bin/mount -n -o remount,rw /
check_status

echo > /etc/mtab
/bin/mount -f -o remount,rw /

echo -n "Mounting proc file system..."
/bin/mount proc
check_status

# End /etc/init.d/mountfs
```

Creating the umountfs script

Create a new file /etc/init.d/umountfs containing the following:

```
#!/bin/sh
# Begin /etc/init.d/umountfs

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

echo -n "Deactivating swap..."
/sbin/swapoff -a
check_status

echo -n "Unmounting file systems..."
/bin/umount -a -r
check_status

# End /etc/init.d/umountfs
```

Creating the sendsignals script

Create a new file /etc/init.d/sendsignals containing the following:

```
#!/bin/sh
# Begin /etc/init.d/sendsignals

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

echo -n "Sending all processes the TERM signal..."
/sbin/killall5 -15
check_status

echo -n "Sending all processes the KILL signal..."
/sbin/killall5 -9
check_status

# End /etc/init.d/sendsignals
```

Creating the checkroot script

Create a new file `/etc/init.d/checkroot` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/checkroot

echo −n "Activating swap..."
/sbin/swapon −a

if [ −f /fastboot ]
then
  echo "Fast boot, no file system check"
else
  /bin/mount −n −o remount,ro /
  if [ $? = 0 ]
  then
    if [ −f /forcecheck ]
    then
      force="−f"
    else
      force=""
    fi

    echo "Checking root file system..."
    /sbin/fsck $force −a /

    if [ $? −gt 1 ]
    then
      echo
      echo "fsck failed. Please repair your file system manually by"
      echo "running fsck without the −a option"
      echo
      echo "Please note that the file system is currently mounted in"
      echo "read−only mode."
      echo
      echo "I will start sulogin now. CTRL+D will reboot your system."
      /sbin/sulogin
      /sbin/reboot −f
    fi
  else
    echo "Cannot check root file system because it is not mounted in"
    echo "read−only mode."
  fi
fi

# End /etc/init.d/checkroot
```

Creating the setclock script

Create a new file `/etc/init.d/setclock` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/setclock

check_status()
{
  if [ $? = 0 ]
  then echo ""
  else
    echo "FAILED"
  fi
}

echo -n "Setting clock..."
/sbin/hwclock
check_status

# End /etc/init.d/setclock
```

Creating the sysklogd script

Create a new file `/etc/init.d/sysklogd` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/sysklogd

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

case "$1" in
  start)
    echo −n "Starting system log daemon..."
    start−stop−daemon −S −q −o −x /usr/sbin/syslogd −− −m 0
    check_status

    echo −n "Starting kernel log daemon..."
    start−stop−daemon −S −q −o −x /usr/sbin/klogd
    check_status
    ;;

  stop)
    echo −n "Stopping kernel log daemon..."
    start−stop−daemon −K −q −o −p  /var/run/klogd.pid
    check_status

    echo −n "Stopping system log daemon..."
    start−stop−daemon −K −q −o −p /var/run/syslogd.pid
    check_status
    ;;

  reload)
    echo −n "Reloading system load daemon configuration file..."
    start−stop−daemon −K −q −o −s 1 −p /var/run/syslogd.pid
    check_status
    ;;

  restart)
    echo −n "Stopping kernel log daemon..."
    start−stop−daemon −K −q −o −p /var/run/klogd.pid
```

```
    check_status

    echo −n "Stopping system log daemon..."
    start−stop−daemon −K −q −o −p /var/run/syslogd.pid
    check_status

    sleep 1

    echo −n "Starting system log daemon..."
    start−stop−daemon −S −q −o −x /usr/sbin/syslogd −− −m 0
    check_status

    echo −n "Starting kernel log daemon..."
    start−stop−daemon −S −q −o −x /usr/sbin/klogd
    check_status
    ;;

  *)
    echo "Usage: $0 {start|stop|reload|restart}"
    exit 1
    ;;
esac

# End /etc/init.d/sysklogd
```

Setting up symlinks and permissions

Give these files the proper permissions and create the necessary symlinks by running the following commands:

```
root:~#   cd /etc/init.d
root:/etc/init.d#   chmod 755 rcS reboot halt mountfs umountfs
root:/etc/init.d#   chmod 755 sendsignals checkroot setclock
sysklogd
root:/etc/init.d#   cd ../rc0.d
root:/etc/rc0.d#   ln -s ../init.d/sysklogd K90sysklogd
root:/etc/rc0.d#   ln -s ../init.d/sendsignals S80sendsignals
root:/etc/rc0.d#   ln -s ../init.d/umountfs S90umountfs
root:/etc/rc0.d#   ln -s ../init.d/halt S99halt
root:/etc/rc0.d#   cd ../rc6.d
root:/etc/rc6.d#   ln -s ../init.d/sysklogd K90sysklogd
root:/etc/rc6.d#   ln -s ../init.d/sendsignals S80sendsignals
root:/etc/rc6.d#   ln -s ../init.d/umountfs S90umountfs
root:/etc/rc6.d#   ln -s ../init.d/reboot S99reboot
root:/etc/rc6.d#   cd ../rcS.d
root:/etc/rcS.d#   ln -s ../init.d/setclock S01setclock
root:/etc/rcS.d#   ln -s ../init.d/checkroot S05checkroot
root:/etc/rcS.d#   ln -s ../init.d/mountfs S10mountfs
root:/etc/rcS.d#   cd /etc/rc2.d
root:/etc/rc2.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:

```
# Begin /etc/fstab

/dev/<LFS−partition designation> / ext2 defaults 0 1
/dev/<swap−partition designation> none swap sw 0 0
proc /proc proc defaults 0 0

# End /etc/fstab
```

Replace <LFS−partition designation> and <swap−partition designation> with the appropriate devices (/dev/sda5 and /dev/hda6 in my case).

Chapter 13. Setting up basic networking

Installing network software

Installing Netkit−base

Install Netkit−base by running the following commands:

```
root:/usr/src/netkit-base-0.17...#   ./configure --prefix=/usr
root:/usr/src/netkit-base-0.17...#   make
root:/usr/src/netkit-base-0.17...#   make install
root:/usr/src/netkit-base-0.17...#   cd etc.sample
root:/usr/src/netkit-base-0.17.../etc.sample#   cp services \
> protocols /etc
```

Installing Net−tools

Install Net−tools by running the following commands:

```
root:/usr/src/net-tools-1.54#   make
root:/usr/src/net-tools-1.54#   make install
```

Creating network boot scripts

Creating the /etc/init.d/localnet bootscript

Create a new file `/etc/init.d/localnet` containing the following:

```
#!/bin/sh
# Begin /etc/init.d/localnet

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

echo -n "Setting up loopback device..."
/sbin/ifconfig lo 127.0.0.1
check_status

echo -n "Setting up hostname..."
/bin/hostname --file /etc/hostname
check_status

# End /etc/init.d/localnet
```

Setting up permissions and symlink

Set the proper file permissions and create the necessary symlink by running the following commands:

```
root:~#   cd /etc/init.d
root:/etc/init.d#   chmod 755 /etc/init.d/localnet
root:/etc/init.d#   cd ../rcS.d
root:/etc/rcS.d#   ln -s ../init.d/localnet S03localnet
```

Creating the /etc/hostname file

Create a new file `/etc/hostname` and put the hostname in it. This is not the FQDN (Fully Qualified Domain Name). This is the name you wish to call your computer in a network. An example:

```
# Begin /etc/hostname

lfs

# End /etc/hostname
```

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` containing:

```
# Begin /etc/hosts (no network card version)
```

```
127.0.0.1 www.linuxfromscratch.org <contents of /etc/hostname> localhost

# End /etc/hosts (no network card version)
```

If you do configure a network card, create a new file `/etc/hosts` containing:

```
# Begin /etc/hosts (network card version)

127.0.0.1 localhost
192.168.1.1 www.linuxfromscratch.org <contents of /etc/hostname>

# End /etc/hosts (network card version)
```

Of course, change the 192.168.1.1 and www.linuxfromscratch.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 file

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:

```
#!/bin/sh
# Begin /etc/init.d/ethnet

check_status()
{
  if [ $? = 0 ]
  then
    echo "OK"
  else
    echo "FAILED"
  fi
}

IPADDR="209.83.245.12" # Replace with your own IP address
NETMAKSK="255.255.255.0" # Replace with your own Netmask
NETWORK="209.83.245.0" # Replace with your own Network address
BROADCAST="209.83.245.255" # Replace with your own Broadcast addr.
GATEWAY="209.83.245.1" # Replace with your own Gateway address
```

```
echo -n "Setting up eth0..."
/sbin/ifconfig eth0 $(IPADDR) broadcast $(BROADCAST) netmask $(NETMASK)
check_status

echo "Setting up route..."
/sbin/route add -net $(NETWORK) netmask $(NETMASK) eth0
check_status

echo "Adding default gateway..."
/sbin/route add default gw $(GATEWAY) metric 1
check_status

# End /etc/init.d/ethnet
```

Setting up permissions and symlink

Set the proper file permissions and create the necessary symlink by running the following commands:

```
root:~#   cd /etc/init.d
root:/etc/init.d#   chmod 755 /etc/init.d/ethnet
root:/etc/init.d#   cd ../rc2.d
root:/etc/rc2.d#   ln -s ../init.d/ethnet S10ethnet
```

Chapter 14. Making the LFS system bootable

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:

If you need to apply the Kernel USB patch, do that by running the following commands:

```
root:~#   cd /usr/src/linux
root:/usr/src/linux#   gzip -dc
../usb-2.3.50-1-for-2.2.14.diff.gz | patch -p1
```

The build the actual kernel, run the following commands:

```
root:/usr/src/linux#   make pmac_config
root:/usr/src/linux#   make menuconfig
root:/usr/src/linux#   make dep
root:/usr/src/linux#   make vmlinux
root:/usr/src/linux#   cp System.map /boot
root:/usr/src/linux#   cp vmlinux /boot/lfskernel
```

Updating BootX

Now we have to get /boot/lfskernel to the Mac OS side so we can boot our LFS system. There are a few ways to copy the /boot/lfskernel file to the Linux kernel folder on the Mac OS side.

The easiest way is be to mount a Mac HFS partition under Linux and copy the kernel to that partition in the right folder. The Linux kernel currently does not support the HFS+ partition, do do not attempt to mount a Mac HFS+ (also known as HFS Extended) partition under Linux.

Copy the kernel to your Mac HFS partition by running the following commands:

```
root:~#   mkdir /mnt/exchange
root:~#   mount -t hfs /dev/sda1 /mnt/exchange
root:~#   cp /boot/lfskernel /mnt/exchange
root:~#   umount /dev/sda1
```

Of course, replace /dev/sda1 by your Mac partition's designation.

If you can't mount the Mac partition for some reason (for example because it's a HFS+ partition) you'll have to email the kernel to yourself. Use a shell on your normal Linux's system (not the the chroot'ed environment) to obtain the kernel image. Compress it with gzip and attach it to an email. Boot into your MacOS and download the email. You can use the MacGzip application to ungzip the kernel image and move it to the "Linux Kernels" folder under "System Folder". If you don't have MacGzip installed, you can download it from http://macinsearch.com/infomac/cmp/mac−gzip−111.html

Of course, if the kernel is small enough to fit on a floppy disk, and your Mac has a floppy drive, you can transfer it that way. Of if you have a ZIP drive at your disposal, you can transfer it on that medium.

Testing the system

Now that all software has been installed, bootscripts have been written and the local network is setup, it's time for you to reboot your computer and test these new scripts to verify that they actually work. You first want to execute them manually from the /etc/init.d directory so you can fix the most obvious problems (typos, wrong paths and such). When those scripts seem to work just fine manually they should also work during a system start or shutdown. There's only one way to test that. 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).

At this point your basic LFS system is ready for use. Everything else that follows now is optional, so you can skip packages at your own discretion. But do keep in mind that if you skip packages (especially libraries) you can break dependencies of other packages. For example, when the Lynx browser is installed, the zlib library is installed as well. You can decide to skip the zlib library, but this library isn't used by Lynx alone. Other packages require this library too. The same may apply to other libraries and programs.

IV. Part IV – Appendixes

Table of Contents

Appendix A. Resources

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

- Sendmail published by O'Reilly. ISBN: 1−56592−222−0
- 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

- Linux Network Administrator's Guide online at http://www.linuxdoc.org
- ISP−Hookup−HOWTO online at http://www.linuxdoc.org

Other

- The various man and info pages that come with the packages