The Linux Bootdisk HOWTO

Graham Chapman, `grahamc@zeta.org.au`

v1.01, 6 February 1995

This document describes how to create Linux boot, boot/root and utility maintenance disks. These disks could be used as rescue disks or to test new kernels.

Contents

1 Introduction

1.1 Why Build Boot Disks?

Linux boot disks are useful in a number of situations, such as:

- Testing a new kernel.
- Recovering from disk or system failure. Such a failure could be anything from a lost boot sector to a disk head crash.

There are several ways of producing boot disks:

- Use one from a distribution such as Slackware. This will at least allow you to boot.
- Use a rescue package to set up disks designed to be used as rescue disks.
- Learn what is required for each of the various types of disk to operate, then build your own.

I choose the last option - learn how it works so that you can do it yourself. That way, if something breaks, you can work out what to do to fix it. Plus you learn a lot about how Linux works along the way.

Experienced Linux users may find little of use in this document. However users new to Linux system administration who wish to protect against root disk loss and other mishaps may find it useful.

A note on versions - this document has been updated to support the following packages and versions:

- Linux 1.1.73
- LILO 0.15

Copyright (c) Graham Chapman 1995.

Permission is granted for this material to be freely used and distributed, provided the source is acknowledged. No warranty of any kind is provided. You use this material at your own risk.

1.2 Feedback and Credits

I welcome any feedback, good or bad, on the content of this document. Please let me know if you find any errors or omissions.

I thank the following people for correcting errors and providing useful suggestions for improvement:

```
Randolph Bentson
Bjxrn-Helge Mevik
Johannes Stille
```

1.3 Change History

v1.01, 6 February 1995

- Fix: DO NOT cp <kernel file> /dev/fd0 - this will overwrite any file system on the diskette.
- Fix: Put LILO boot.b and map files on target disk.

- Add: -dp flags to cp commands to avoid problems.
- Chg: restructure to try to improve readability.
- Add: can now use ext2 filesystem on root diskettes.
- Chg: can now separate boot and root diskettes.
- Add: credits section in Introduction.
- Add: FAQ.

v1.0, 2 January 1995

- Converted to conform to HOWTO documentation standards.
- Added new section - Change History.
- Various minor corrections.

v0.10, 1 November 1994 Original version, labelled "draft".

2 Disks

2.1 Summary of Disk Types

I classify boot-related disks into 4 types. The discussion here and throughout this document uses the term "disk" to refer to diskettes unless otherwise specified. Most of the discussion could be equally well applied to hard disks.

A summary of disk types and uses is:

boot

A disk containing a kernel which can be booted. The disk can contain a filesystem and use a boot loader to boot, or it can simply contain the kernel only at the start of the disk. The disk can be used to boot the kernel using a root file system on another disk. This could be useful if you lost your boot loader due to, for example, an incorrect installation attempt.

root

A disk with a file system containing everything required to run a Linux system. It does not necessarily contain either a kernel or a boot loader.

This disk can be used to run the system independently of any other disks, once the kernel has been booted. A special kernel feature allows a separate root disk to be mounted after booting, with the root disk being automatically copied to a ramdisk.

You could use this type of disk to check another disk for corruption without mounting it, or to restore another disk following disk failure or loss of files.

boot/root

A disk which is the same as a root disk, but contains a kernel and a boot loader. It can be used to boot from, and to run the system. The advantage of this type of disk is that is it compact - everything required is on a single disk. However the gradually increasing size of everything means that it won't necessarily always be possbile to fit everything on a single diskette.

utility

A disk which contains a file system, but is not intended to be mounted as a root file system. It is an additional data disk. You would use this type of disk to carry additional utilities where you have too much to fit on your root disk.

The term "utility" only really applies to diskettes, where you would use a utility disk to store additional recovery utility software.

2.2 Boot

2.2.1 Overview

All PC systems start the boot process by executing code in ROM to load the sector from sector 0, cylinder 0 of the boot drive and try and execute it. On most bootable disks, sector 0, cylinder 0 contains either:

- code from a boot loader such as LILO, which locates the kernel, loads it and executes it to start the boot proper.
- the start of an operating system kernel, such as Linux.

If a Linux kernel has been written to a diskette as a raw device, then the first sector will be the first sector of the Linux kernel itself, and this sector will continue the boot process by loading the rest of the kernel and running Linux. For a more detailed description of the boot sector contents, see the documentation in lilo-01.5 or higher.

An alternative method of storing a kernel on a boot disk is to create a filesystem, not as a root filesystem, but simply as a means of installing LILO and thus allowing boot-time command line options to be specified. For example, the same kernel could then be used to boot using a hard disk root filesystem, or a diskette root filesystem. This could be useful if you were trying to rebuild the hard disk filesystem, and wanted to repeatedly test results.

2.2.2 Setting Pointer to Root

The kernel must somehow obtain a pointer to the drive and partititon to be mounted as the root drive. This can be provided in several ways:

- By setting `ROOT_DEV = <device>` in the Linux kernel makefile and rebuilding the kernel (for advice on how to rebuild the kernel, read the Linux FAQ and look in `/usr/src/linux`). Comments in the Linux makefile describe the valid values for `<device>`.

- By running the rdev utility:

  ```
  rdev {$<$}filename{$>$} {$<$}device{$>$}
  ```

  This will set the root device of the kernel contained in <`filename`> to be <`device`>. For example:

  ```
  rdev Image /dev/sda1
  ```

  This sets the root device in the kernel in Image to the first partition on the first SCSI drive.

There are some alternative ways of issuing the rdev command. Try:

```
rdev -?
```

and it will display command usage.

There is usually no need to configure the root device for boot diskette use, because the kernel currently used to boot from probably already points to the root drive device. The need can arise, howoever, if you obtain a kernel from another machine, for example, from a distribution, or if you want to use the kernel to boot a root diskette. It never hurts to check, though. To use rdev to check the current root device in a kernel file, enter the command:

```
rdev -r <filename>
```

It is possible to change the root device set in a kernel by means other than using rdev. For details, see the FAQ at the end of this document.

2.2.3 Copying Kernel to Boot Diskette

Once the kernel has been configured then it must be copied to the boot diskette.

If the disk is not intended to contain a file system, then the kernel must be copied using the dd command, as follows:

```
dd if={$<$}filename{$>$} of={$<$}device{$>$}

where   {$<$}filename{$>$} is the name of the kernel
and     {$<$}device{$>$} is the diskette raw device,
        usually /dev/fd0
```

The seek parameter to the dd command should NOT be used. The file must be copied to start at the boot sector (sector 0, cylinder 0), and omitting the seek parameter will do this.

The output device name varies. Many systems have `/dev/fd0` as an alias of one sort or another for the "real" device name for the default diskette drive. For example, where the default drive (i.e. "drive A:" in DOS) is a high density 3 1/2 inch diskette drive, the device name will be `/dev/fd0H1440`, but usually `/dev/fd0` points to the same device.

Where the kernel is to be copied to a boot disk containing a filesystem, then the disk is mounted at a suitable point in a currently-mounted filesystem, then the cp command is used. For example:

```
mount -t ext2 /dev/fd0 /mnt
cp Image /mnt
umount /mnt
```

2.3 Root

2.3.1 Overview

A root disk contains a complete working Linux system, but without necessarily including a kernel. In other words, the disk may not be bootable, but once the kernel is running, the root disk contains everything needed to support a full Linux system. To be able to do this, the disk must include the minimum requirements for a Linux system:

- File system.
- Minimum set of directories - dev, proc, bin, etc, lib, usr, tmp.
- Basic set of utilities - bash (to run a shell), ls, cp etc.
- Minimum set of config files - rc, inittab, fstab etc.
- Runtime library to provide basic functions used by utilities.

Of course, any system only becomes useful when you can run something on it, and a root diskette usually only becomes useful when you can do something like:

- Check a file system on another drive, for example to check your root file system on your hard drive, you need to be able to boot Linux from another drive, as you can with a root diskette system. Then you can run fsck on your original root drive while it is not mounted.
- Restore all or part of your original root drive from backup using archive/compression utilities including cpio, tar, gzip and ftape.

2.4 Boot/Root

This is essentially the same as the root disk, with the addition of a kernel and a boot loader such as LILO.

With this configuration, a kernel file is copied to the root file system, and LILO is then run to install a configuration which points to the kernel file on the target disk. At boot time, LILO will boot the kernel from the target disk.

Several files must be copied to the diskette for this method to work. Details of these files and the required LILO configuration, including a working sample, are given below in the section titled "LILO".

2.4.1 RAM Drives and Root Filesystems on Diskette

For a diskette root filesystem to be efficient, you need to be able to run it from a ramdrive, i.e. an emulated disk drive in main memory. This avoids having the system run at a snail's pace, which a diskette would impose.

There is an added benefit from using a ramdrive - the Linux kernel includes an automatic ramdisk root feature, whereby it will, under certain circumstances, automatically copy the contents of a root diskette to a RAM disk, and then switch the root drive to be the RAM disk instead of the diskette. This has two major benefits:

- The system runs a lot faster.
- The diskette drive is freed up to allow other diskettes to be used on a single-diskette drive system.

The requirements for this feature to be invoked are:

- The file system on the diskette drive must be either a minix or an ext2 file system. The ext2 file system is generally the preferred file system to use. Note that if you have a Linux kernel earlier than 1.1.73, then you should see the comments in the section below titled "File Systems" to see whether your kernel will support ext2. If your kernel is old then you may have to use minix. This will not cause any significant problems.
- A RAM disk must be configured into the kernel, and it must be at least as big as the diskette drive.

A RAM disk can be configured into the kernel in several ways:

- By uncommenting the RAMDISK macro in the Linux kernel makefile, so that it reads:

  ```
  RAMDISK = -DRAMDISK=1440
  ```

  to define a ramdisk of 1440 1K blocks, the size of a high-density diskette.
- By running the rdev utility, available on most Linux systems. This utility displays or sets values for several things in the kernel, including the desired size for a ramdisk. To configure a ramdisk of 1440 blocks into a kernel in a file named Image, enter:

  ```
  rdev -r Image 1440
  ```

  this might change in the future, of course. To see what your version of rdev does, enter the command:

  ```
  rdev -?
  ```

  and it should display its options.
- By using the boot loader package LILO to configure it into your kernel at boot time. This can be done using the LILO configuration parameter:

  ```
  ramdisk = 1440
  ```

  to request a RAM drive of 1440 1K blocks at boot time.

- By interrupting a LILO automatic boot and adding ramdisk=1440 to the command line. For example, such a command line might be:

  ```
  vmlinux ramdisk=1440
  ```

  See the section on LILO for more details.
- By editing the kernel file and altering the values near the start of the file which record the ramdisk size. This is definitely a last resort, but can be done. See the FAQ near the end of this document for more details.

The easiest of these methods is LILO configuration, because you need to set up a LILO configuration file anyway, so why not add the ramdisk size here?

LILO configuration is briefly described in a section titled "LILO" below, but it is advisable to obtain the latest stable version of LILO from your nearest Linux mirror site, and read the documentation that comes with it.

2.5 Utility

Often one disk is not sufficient to hold all the software you need to be able to perform rescue functions of analysing, repairing and restoring corrupted disk drives. By the time you include tar, gzip e2fsck, fdisk, Ftape and so on, there is enough for a whole new diskette, maybe even more if you want lots of tools.

This means that a rescue set often requires a utility diskette, with a file system containing any extra files required. This file system can then be mounted at a convenient point, such as /usr, on the boot/root system.

Creating a file system is fairly easy, and is described above in the section titled "File Systems" above.

3 Components

3.1 File Systems

The Linux kernel now supports two file system types for root disks to be automatically copied to ramdisk. These are minix and ext2, of which ext2 is the preferred file system. The ext2 support was added sometime between 1.1.17 and 1.1.57, I'm not sure exactly which. If you have a kernel within this range then edit /usr/src/linux/drivers/block/ramdisk.c and look for the word "ext2". If it is not found, then you will have to use a minix file system, and therefore the "mkfs" command to create it.

To create an ext2 file system on a diskette on my system, I issue the following command:

```
mke2fs /dev/fd0
```

The mke2fs command will automatically detect the space available and configure itself accordingly. It does not therefore require any parameters.

An easy way to test the result is to create a system using the above command or similar, and then attempt to mount the diskette. If it is an ext2 system, then the command:

```
mount -t ext2 /dev/fd0 /<mount point>
```

should work.

3.2 Kernel

3.2.1 Building a Custom Kernel

In most cases it would be possible to copy your current kernel and boot the diskette from that. However there may be cases where you wish to build a separate one.

One reason is size. The kernel is one of the largest files in a minimum system, so if you want to build a boot/root diskette, then you will have to reduce the size of the kernel as much as possible. The kernel now supports changing the diskette after booting and before mounting root, so it is not necessary any more to squeeze the kernel into the same disk as everything else, therefore these comments apply only if you choose to build a boot/root diskette.

There are two ways of reducing kernel size:

- Building it with the minumum set of facilities necessary to support the desired system. This means leaving out everything you don't need. Networking is a good thing to leave out, as well as support for any disk drives and other devices which you don't need when running your boot/root system.
- Compressing it, using the standard compressed-kernel option included in the makefile:

  ```
  make zImage
  ```

  Refer to the documentation included with the kernel source for up-to-date information on building compressed kernels. Note that the kernel source is usually in /usr/src/linux.

Having worked out a minimum set of facilities to include in a kernel, you then need to work out what to add back in. Probably the most common uses for a boot/root diskette system would be to examine and restore a corrupted root file system, and to do this you may need kernel support.

For example, if your backups are all held on tape using Ftape to access your tape drive, then, if you lose your current root drive and drives containing Ftape, then you will not be able to restore from your backup tapes. You will have to reinstall Linux, download and reinstall Ftape, and then try and read your backups.

It is probably desirable to maintain a copy of the same version of backup utilities used to write the backups, so that you don't waste time trying to install versions that cannot read your backup tapes.

The point here is that, whatever I/O support you have added to your kernel to support backups should also be added into your boot/root kernel. Note, though, that the Ftape module (or at least the one I have) is quite large and will not fit on your boot/root diskette. You will need to put it on a utility diskette - this is described below in the section titled "ADDING UTILITY DISKETTES".

The procedure for actually building the kernel is described in the documentation that comes with the kernel. It is quite easy to follow, so start by looking in /usr/src/linux. Note that if you have trouble building a kernel, then you should probably not attempt to build boot/root systems anyway.

3.3 Devices

A /dev directory containing a special file for all devices to be used by the system is mandatory for any Linux system. The directory itself is a normal directory, and can be created with the mkdir command in the normal way. The device special files, however, must be created in a special way, using the mknod command.

There is a shortcut, though - copy your existing /dev directory contents, and delete the ones you don't want. The only requirement is that you copy the device special files using the -R option. This will copy the directory without attempting to copy the contents of the files. Note that if you use lower caser, as in "-r", there will be a vast difference, because you will probably end up copying the entire contents of all of your hard disks - or at least as much of them as will fit on a diskette! Therefore, take care, and use the command:

```
cp -dpR /dev /mnt
```

assuming that the diskette is mounted at /mnt. The dp switches ensure that symbolic links are copied as links (rather than the target file being copied) and that the original file attributes are preserved, thus preserving ownership information.

If you want to do it the hard way, use ls -l to display the major and minor device numbers for the devices you want, and create them on the diskette using mknod.

Many distributions include a shell script called MAKEDEV in the /dev directory. This shell script could be used to create the devices, but it is probably easier to just copy your existing ones, especially for rescue disk purposes.

3.4 Directories

It might be possible to get away with just /dev, /proc and /etc to run a Linux system. I don't know - I've never tested it. However a reasonable minimum set of directories consists of the following:

/dev

Required to perform I/O with devices

/proc

Required by the ps command

/etc

System configuration files

/bin

Utility executables considered part of the system

/lib

Shared libraries to provide run-time support

/mnt

A mount point for maintenance on other disks

/usr

Additional utilities and applications

Note that the directory tree presented here is for root diskette use only. Refer to the Linux File System Standard for much better information on how file systems should be structured in "standard" Linux systems.

Four of these directories can be created very easily:

- /dev is described above in the section titled DEVICES.
- /proc only needs to exist. Once the directory is created using mkdir, nothing more is required.
- Of the others, /mnt and /usr are included in this list only as mount points for use after the boot/root system is running. Hence again, these directories only need to be created.

The remaining 3 directories are described in the following sections.

3.4.1 /etc

This directory must contain a number of configuration files. On most systems, these can be divided into 3 groups:

- Required at all times, e.g. rc, fstab, passwd.
- May be required, but no-one is too sure.
- Junk that crept in.

Files which are not essential can be identified with the command:

```
ls -ltru
```

This lists files in reverse order of date last accessed, so if any files are not being accessed, then they can be omitted from a root diskette.

On my root diskettes, I have the number of config files down to 15. This reduces my work to dealing with three sets of files:

- The ones I must configure for a boot/root system:

```
rc      system startup script
fstab   list of file systems to be mounted
inittab parameters for the init process - the
        first process started at boot time.
```

- the ones I should tidy up for a boot/root system:

```
passwd  list of logins
shadow  contains passwords
```

These should be pruned on secure systems to avoid copying user's passwords off the system, and so that when you boot from diskette, unwanted logins are rejected.

- The rest. They work at the moment, so I leave them alone.

Out of this, I only really have to configure two files, and what they should contain is suprisingly small.

- rc should contain:

  ```
  #!/bin/sh
  /etc/mount -av
  /bin/hostname boot_root
  ```

  and I don't really need to run hostname - it just looks nicer if I do. Even mount is actually only needed to mount /proc to support the ps command - Linux will run without it.

- fstab should contain:

  ```
  /dev/fd0        /               ext2    defaults
  /proc           /proc           proc    defaults
  ```

  I don't think that the first entry is really needed, but I find that if I leave it out, mount won't mount /proc.

Inittab should be ok as is, unless you want to ensure that users on serial ports cannot login. To prevent this, comment out all the entries for /etc/getty which include a ttys or ttyS device at the end of the line. Leave in the tty ports so that you can login at the console.

For the rest, just copy all the text files in your /etc directory, plus all the executables in your /etc directory that you cannot be sure you do not need. As a guide, consult the sample ls listing in "Sample Boot/Root ls-lR Directory Listing" - this is what I have, so probably it will be sufficient for you if you copy only those files.

3.4.2 /bin

Here is a convenient point to place the extra utilities you need to perform basic operations, utilities such as ls, mv, cat, dd etc.

See the section titled "Sample Boot/Root ls-lR Directory Listing" for the list of files that I place in my boot/root /bin directory. You may notice that it does not include any of the utilities required to restore from backup, such as cpio, tar, gzip etc. That is because I place these on a separate utility diskette, to save space on the boot/root diskette. Once I have booted my boot/root diskette, it then copies itself to the ramdisk leaving the diskette drive free to mount another diskette, the utility diskette. I usually mount this as /usr.

Creation of a utility diskette is described below in the section titled "Adding Utility Diskettes".

3.4.3 /lib

Two libraries are required to run many facilities under Linux:

- ld.so
- libc.so.4

If they are not found in your /lib directory then the system will be unable to boot. If you're lucky you may see an error message telling you why.

These should be present in you existing /lib directory. Note that libc.so.4 may be a symlink to a libc library with version number in the filename. If you issue the command:

```
ls -l /lib
```

you will see something like:

```
libc.so.4 -> libc.so.4.5.21
```

In this case, the libc library you want is libc.so.4.5.21.

3.5 LILO

3.5.1 Overview

For the boot/root to be any use, it must be bootable. To achieve this, the easiest way (possibly the only way?) is to install a boot loader, which is a piece of executable code stored at sector 0, cylinder 0 of the diskette. See the section above titled "BOOT DISKETTE" for an overview of the boot process.

LILO is a tried and trusted boot loader available from any Linux mirror site. It allows you to configure the boot loader, including:

- Which device is to be mounted as the root drive.
- Whether to use a ramdisk.

3.5.2 Sample LILO Configuration

This provides a very convenient place to specify to the kernel how it should boot. My root/boot LILO configuration file, used with LILO 0.15, is:

```
boot = /dev/fd0
install = ./mnt/boot.b
map = ./mnt/lilo.map
delay = 50
message = ./mnt/lilo.msg
```

```
timeout = 150
compact
image = ./mnt/vmlinux
        ramdisk = 1440
        root = /dev/fd0
```

Note that boot.b, lilo.msg and the kernel must first have been copied to the diskette using a command similar to:

```
cp /boot/boot.b ./mnt
```

If this is not done, then LILO will not run correctly at boot time if the hard disk is not available, and there is little point setting up a rescue disk which requires a hard disk in order to boot.

I run lilo using the command:

```
/sbin/lilo -C <configfile>
```

I run it from the directory containing the mnt directory where I have mounted the diskette. This means that I am telling LILO to install a boot loader on the boot device (/dev/fd0 in this case), to boot a kernel in the root directory of the diskette.

I have also specified that I want the root device to be the diskette, and I want a RAM disk created of 1440 1K blocks, the same size as the diskette. Since I have created an ext2 file system on the diskette, this completes all the conditions required for Linux to automatically switch the root device to the ramdisk, and copy the diskette contents there as well.

The ramdisk features of Linux are described further in the section above titled "RAM DRIVES AND BOOT/ROOT SYSTEMS".

It is also worth considering using the "single" parameter to cause Linux to boot in single-user mode. This could be useful to prevent users logging in on serial ports.

I also use the "DELAY" "MESSAGE" and "TIMEOUT" statements so that when I boot the disk, LILO will give me the opportunity to enter command line options if I wish. I don't need them at present, but I never know when I might want to set a different root device or mount a filesystem read-only.

The message file I use contains the message:

```
Linux Boot/Root Diskette
========================

Enter a command line of the form:

      vmlinux [ command-line options]

If nothing is entered, linux will be loaded with
defaults after 15 seconds.
```

This is simply a reminder to myself what my choices are.

Readers are urged to read the LILO documentation carefully before atttempting to install anything. It is relatively easy to destroy partitions if you use the wrong "boot = " parameter. If you are inexperienced, do NOT run LILO until you are sure you understand it and you have triple-checked your parameters.

3.5.3 Removing LILO

One other thing I might as well add here while I'm on the LILO topic: if you mess up lilo on a drive containing DOS, you can always replace the boot sector with the DOS boot loader by issuing the DOS command:

```
FDISK /MBR
```

where MBR stands for "Master Boot Record". Note that some purists disagree with this, and they may have grounds, but it works.

3.5.4 Useful LILO Options

LILO has several useful options which are worth keeping in mind when building boot disks:

- Command line options - you can enter command line options to set the root device, ramdrive size, special device parameters, or other things. If you include the DELAY = nn statement in your LILO configuration file, then LILO will pause to allow you to select a kernel image to boot, and to enter, on the same line, any options. For example:

  ```
  vmlinux aha152x=0x340,11,3,1 ro
  ```

  will pass the aha152x parameters through to the aha152x scsi disk driver (provided that driver has been included when the kernel was built) and will ask for the root filesystem to be mounted read-only.

- Command line "lock" option - this option asks LILO to store the command line entered as the default command line to be used for all future boots. This is particularly useful where you have a device which cannot be autoselected. By using "lock" you can avoid having to type in the device parameter string every time you boot. For example:

  ```
  vmlinux aha152x=0x340,11,3,1 root=/dev/sda8 ro lock
  ```

- APPEND configuration statement - this allows device parameter strings to be stored in the configuration, as an alternative to using the "lock" command line option. Note that any keywords of the form word=value MUST be enclosed in quotes. For example:

  ```
  APPEND = "aha152x=0x340,11,3,1"
  ```

- DELAY configuration statement - this pauses for DELAY tenths of seconds and allows the user to interrupt the automatic boot of the default command line, so that the user can enter an alternate command line.

4 Samples

4.1 Disk Directory Listings

This lists the contents of files and directories that I keep on my hard disk to use when building boot/root and utility diskettes. It shows which files I put in the /etc and /bin directories on my diskettes.

The sample shell scripts in the next section use these directories and files as a model to build the diskettes.

4.1.1 Boot/Root Disk ls-lR Directory Listing

The boot/root listing is of directory boot_disk:

```
total 226
drwxr-xr-x   2 root     root         1024 Oct  8 13:40 bin/
drwxr-xr-x   2 root     root         3072 Sep  8 16:37 dev/
drwxr-xr-x   2 root     root         1024 Oct  8 12:38 etc/
drwxr-xr-x   2 root     root         1024 Sep 10 14:58 lib/
-rw-r--r--   1 root     root       297956 Jan 25 21:55 vmlinux

boot_disk/bin:
total 366
-rwxr-xr-x   1 root     root         4376 Sep  9 21:34 cat*
-rwxr-xr-x   1 root     root         4112 Sep  9 21:34 chown*
-rwxr-xr-x   1 root     root        12148 Sep  9 21:34 cp*
-rwxr-xr-x   1 root     root         4376 Sep  9 21:34 cut*
-rwxr-xr-x   1 root     root         7660 Sep  9 21:34 dd*
-rwxr-xr-x   1 root     root         4696 Sep  9 21:34 df*
-rwx--x--x   1 root     root         1392 Sep 10 14:13 hostname*
-rwxr-xr-x   1 root     root         5252 Sep  9 21:34 ln*
-rwsr-xr-x   1 root     root         6636 Sep  9 21:34 login*
-rwxr-xr-x   1 root     root        13252 Sep  9 21:34 ls*
-rwxr-xr-x   1 root     root         4104 Sep  9 21:34 mkdir*
-rwxr-xr-x   1 root     root        21504 Sep 10 15:27 more*
-rwxr-xr-x   1 root     root         6744 Sep  9 21:34 mv*
-rwxr-xr-x   1 root     root         9780 Sep  9 21:34 ps*
-rwxr-xr-x   1 root     root         5076 Sep  9 21:34 rm*
-r-xr-xr-x   1 root     root        12604 Sep  9 21:34 sed*
-rwxr-xr-x   1 root     root       222208 Sep  9 21:34 sh*
-rws--x--x   1 root     root        16464 Sep  9 21:34 su*
-rwxr-xr-x   1 root     root         1216 Sep  9 21:34 sync*

boot_disk/dev:
total 73
-rwxr-xr-x   1 root     root         8331 Sep  8 16:31 MAKEDEV*
crw-r--r--   1 root     root      10,   3 Sep  8 16:31 bmouseatixl
crw-r--r--   1 root     root      10,   0 Sep  8 16:31 bmouselogitec
```

```
crw-r--r--   1 root     root      10,   2 Sep  8 16:31 bmousems
crw-r--r--   1 root     root      10,   1 Sep  8 16:31 bmouseps2
crw-------   1 root     root       0,   0 Sep  8 16:31 boot0
crw-r--r--   1 root     root       4,   0 Sep  8 16:31 console
crw-r--r--   1 root     root       5,  64 Sep  8 16:31 cua0
crw-r--r--   1 root     root       5,  65 Sep  8 16:31 cua1
crw-r--r--   1 root     root       5,  66 Sep  8 16:31 cua2
crw-r--r--   1 root     root       5,  67 Sep  8 16:31 cua3
brw-r--r--   1 root     root       2,   0 Sep  8 16:31 fd0
brw-r--r--   1 root     root       2,  12 Sep  8 16:31 fd0D360
brw-r--r--   1 root     root       2,  16 Sep  8 16:31 fd0D720
brw-r--r--   1 root     root       2,  28 Sep  8 16:31 fd0H1440
brw-r--r--   1 root     root       2,  12 Sep  8 16:31 fd0H360
brw-r--r--   1 root     root       2,  16 Sep  8 16:31 fd0H720
brw-r--r--   1 root     root       2,  16 Sep  8 16:31 fd0Q720
brw-r--r--   1 root     root       2,   4 Sep  8 16:31 fd0d360
brw-r--r--   1 root     root       2,   8 Sep  8 16:31 fd0h1200
brw-r--r--   1 root     root       2,  20 Sep  8 16:31 fd0h360
brw-r--r--   1 root     root       2,  24 Sep  8 16:31 fd0h720
brw-r--r--   1 root     root       2,  24 Sep  8 16:31 fd0q720
brw-r--r--   1 root     root       2,   1 Sep  8 16:31 fd1
brw-r--r--   1 root     root       2,  13 Sep  8 16:31 fd1D360
brw-r--r--   1 root     root       2,  17 Sep  8 16:31 fd1D720
brw-r--r--   1 root     root       2,  29 Sep  8 16:31 fd1H1440
brw-------   1 root     root       2,  31 Sep  8 16:31 fd1H1722
brw-r--r--   1 root     root       2,  13 Sep  8 16:31 fd1H360
brw-r--r--   1 root     root       2,  17 Sep  8 16:31 fd1H720
brw-r--r--   1 root     root       2,  17 Sep  8 16:31 fd1Q720
brw-r--r--   1 root     root       2,   5 Sep  8 16:31 fd1d360
brw-r--r--   1 root     root       2,   9 Sep  8 16:31 fd1h1200
brw-r--r--   1 root     root       2,  21 Sep  8 16:31 fd1h360
brw-r--r--   1 root     root       2,  25 Sep  8 16:31 fd1h720
brw-r--r--   1 root     root       2,  25 Sep  8 16:31 fd1q720
brw-r-----   1 root     root       3,   0 Sep  8 16:31 hda
brw-r-----   1 root     root       3,   1 Sep  8 16:31 hda1
brw-r-----   1 root     root       3,   2 Sep  8 16:31 hda2
brw-r-----   1 root     root       3,   3 Sep  8 16:31 hda3
brw-r-----   1 root     root       3,   4 Sep  8 16:31 hda4
brw-r-----   1 root     root       3,   5 Sep  8 16:31 hda5
brw-r-----   1 root     root       3,   6 Sep  8 16:31 hda6
brw-r-----   1 root     root       3,   7 Sep  8 16:31 hda7
brw-r-----   1 root     root       3,   8 Sep  8 16:31 hda8
brw-r-----   1 root     root       3,  64 Sep  8 16:31 hdb
brw-r-----   1 root     root       3,  65 Sep  8 16:31 hdb1
brw-r-----   1 root     root       3,  66 Sep  8 16:31 hdb2
brw-r-----   1 root     root       3,  67 Sep  8 16:31 hdb3
brw-r-----   1 root     root       3,  68 Sep  8 16:31 hdb4
brw-r-----   1 root     root       3,  69 Sep  8 16:31 hdb5
```

```
brw-r-----   1 root     root       3,  70 Sep  8 16:31 hdb6
brw-r-----   1 root     root       3,  71 Sep  8 16:31 hdb7
brw-r-----   1 root     root       3,  72 Sep  8 16:31 hdb8
crw-r-----   1 root     root       1,   2 Sep  8 16:31 kmem
brw-------   1 root     root      12,   0 Sep  8 16:31 loop0
brw-------   1 root     root      12,   1 Sep  8 16:31 loop1
crw-r--r--   1 root     root       6,   0 Sep  8 16:31 lp0
crw-r--r--   1 root     root       6,   1 Sep  8 16:31 lp1
crw-r--r--   1 root     root       6,   2 Sep  8 16:31 lp2
brw-r--r--   1 root     root      12,   0 Sep  8 16:31 mcd0
crw-r-----   1 root     root       1,   1 Sep  8 16:31 mem
crw-r--r--   1 root     root       5,  65 Sep  8 16:31 modem
crw-r--r--   1 root     root       5,  64 Sep  8 16:31 mouse
crw-r--r--   1 root     root      27,   4 Sep  8 16:31 nrft0
crw-r--r--   1 root     root      27,   5 Sep  8 16:31 nrft1
crw-r--r--   1 root     root      27,   6 Sep  8 16:31 nrft2
crw-r--r--   1 root     root      27,   7 Sep  8 16:31 nrft3
crw-------   1 root     root       9, 128 Sep  8 16:31 nrmt0
crw-r--r--   1 root     root       1,   3 Sep  8 16:31 null
crw-r-----   1 root     root       6,   0 Sep  8 16:31 par0
crw-r-----   1 root     root       6,   1 Sep  8 16:31 par1
crw-r-----   1 root     root       6,   2 Sep  8 16:31 par2
crw-r-----   1 root     root       1,   4 Sep  8 16:31 port
crw-r--r--   1 root     root      10,   1 Sep  8 16:31 ps2aux
crw-r--r--   1 root     root       4, 128 Sep  8 16:31 ptyp0
crw-r--r--   1 root     root       4, 129 Sep  8 16:31 ptyp1
crw-r--r--   1 root     root       4, 130 Sep  8 16:31 ptyp2
crw-r--r--   1 root     root       4, 131 Sep  8 16:31 ptyp3
crw-r--r--   1 root     root       4, 132 Sep  8 16:31 ptyp4
crw-r--r--   1 root     root       4, 133 Sep  8 16:31 ptyp5
crw-r--r--   1 root     root       4, 134 Sep  8 16:31 ptyp6
crw-r--r--   1 root     root       4, 135 Sep  8 16:31 ptyp7
crw-r--r--   1 root     root       4, 136 Sep  8 16:31 ptyp8
crw-r--r--   1 root     root       4, 137 Sep  8 16:31 ptyp9
crw-r--r--   1 root     root       4, 138 Sep  8 16:31 ptypa
crw-r--r--   1 root     root       4, 139 Sep  8 16:31 ptypb
crw-r--r--   1 root     root       4, 140 Sep  8 16:31 ptypc
crw-r--r--   1 root     root       4, 141 Sep  8 16:31 ptypd
crw-r--r--   1 root     root       4, 142 Sep  8 16:31 ptype
crw-r--r--   1 root     root       4, 143 Sep  8 16:31 ptypf
brw-r-----   1 root     root       1,   0 Sep  8 16:31 ram
crw-r--r--   1 root     root      27,   0 Sep  8 16:31 rft0
crw-r--r--   1 root     root      27,   1 Sep  8 16:31 rft1
crw-r--r--   1 root     root      27,   2 Sep  8 16:31 rft2
crw-r--r--   1 root     root      27,   3 Sep  8 16:31 rft3
crw-------   1 root     root       9,   0 Sep  8 16:31 rmt0
brw-r-----   1 root     root       8,   0 Sep  8 16:31 sda
brw-r-----   1 root     root       8,   1 Sep  8 16:31 sda1
```

```
brw-r-----   1 root     root       8,   2 Sep  8 16:31 sda2
brw-r-----   1 root     root       8,   3 Sep  8 16:31 sda3
brw-r-----   1 root     root       8,   4 Sep  8 16:31 sda4
brw-r-----   1 root     root       8,   5 Sep  8 16:31 sda5
brw-r-----   1 root     root       8,   6 Sep  8 16:31 sda6
brw-r-----   1 root     root       8,   7 Sep  8 16:31 sda7
brw-r-----   1 root     root       8,   8 Sep  8 16:31 sda8
brw-r-----   1 root     root       8,  16 Sep  8 16:31 sdb
brw-r-----   1 root     root       8,  17 Sep  8 16:31 sdb1
brw-r-----   1 root     root       8,  18 Sep  8 16:31 sdb2
brw-r-----   1 root     root       8,  19 Sep  8 16:31 sdb3
brw-r-----   1 root     root       8,  20 Sep  8 16:31 sdb4
brw-r-----   1 root     root       8,  21 Sep  8 16:31 sdb5
brw-r-----   1 root     root       8,  22 Sep  8 16:31 sdb6
brw-r-----   1 root     root       8,  23 Sep  8 16:31 sdb7
brw-r-----   1 root     root       8,  24 Sep  8 16:31 sdb8
brw-------   1 root     root       8,  32 Sep  8 16:31 sdc
brw-------   1 root     root       8,  33 Sep  8 16:31 sdc1
brw-------   1 root     root       8,  34 Sep  8 16:31 sdc2
brw-------   1 root     root       8,  35 Sep  8 16:31 sdc3
brw-------   1 root     root       8,  36 Sep  8 16:31 sdc4
brw-------   1 root     root       8,  37 Sep  8 16:31 sdc5
brw-------   1 root     root       8,  38 Sep  8 16:31 sdc6
brw-------   1 root     root       8,  39 Sep  8 16:31 sdc7
brw-------   1 root     root       8,  40 Sep  8 16:31 sdc8
brw-------   1 root     root       8,  48 Sep  8 16:31 sdd
brw-------   1 root     root       8,  49 Sep  8 16:31 sdd1
brw-------   1 root     root       8,  50 Sep  8 16:31 sdd2
brw-------   1 root     root       8,  51 Sep  8 16:31 sdd3
brw-------   1 root     root       8,  52 Sep  8 16:31 sdd4
brw-------   1 root     root       8,  53 Sep  8 16:31 sdd5
brw-------   1 root     root       8,  54 Sep  8 16:31 sdd6
brw-------   1 root     root       8,  55 Sep  8 16:31 sdd7
brw-------   1 root     root       8,  56 Sep  8 16:31 sdd8
brw-------   1 root     root       8,  64 Sep  8 16:31 sde
brw-------   1 root     root       8,  65 Sep  8 16:31 sde1
brw-------   1 root     root       8,  66 Sep  8 16:31 sde2
brw-------   1 root     root       8,  67 Sep  8 16:31 sde3
brw-------   1 root     root       8,  68 Sep  8 16:31 sde4
brw-------   1 root     root       8,  69 Sep  8 16:31 sde5
brw-------   1 root     root       8,  70 Sep  8 16:31 sde6
brw-------   1 root     root       8,  71 Sep  8 16:31 sde7
brw-------   1 root     root       8,  72 Sep  8 16:31 sde8
brw-r--r--   1 root     root      11,   0 Sep  8 16:31 sr0
brw-r-----   1 root     root      11,   1 Sep  8 16:31 sr1
brw-r-----   1 root     root      11,   2 Sep  8 16:31 sr2
brw-r-----   1 root     root       3,   1 Sep  8 16:31 swap
crw-r--r--   1 root     root       5,   0 Sep  8 16:31 tty
```

```
crw-r--r--   1 root     root       4,   0 Sep  8 16:31 tty0
crw-------   1 root     root       4,   1 Sep  8 16:31 tty1
crw-r--r--   1 root     root       4,   2 Sep  8 16:31 tty2
-rw-r--r--   1 root     root           20 Sep  8 16:31 tty21
crw-r--r--   1 root     root       4,   3 Sep  8 16:31 tty3
crw-r--r--   1 root     root       4,   4 Sep  8 16:31 tty4
crw-r--r--   1 root     root       4,   5 Sep  8 16:31 tty5
crw-r--r--   1 root     root       4,   6 Sep  8 16:31 tty6
crw-------   1 root     root       4,   7 Sep  8 16:31 tty7
crw-------   1 root     root       4,   8 Sep  8 16:31 tty8
crw-r--r--   1 root     root       4,  64 Sep  8 16:31 ttyS0
crw-r--r--   1 root     root       4,  65 Sep  8 16:31 ttyS1
crw-r--r--   1 root     root       4,  66 Sep  8 16:31 ttyS2
crw-r--r--   1 root     root       4, 192 Sep  8 16:31 ttyp0
crw-r--r--   1 root     root       4, 193 Sep  8 16:31 ttyp1
crw-r--r--   1 root     root       4, 194 Sep  8 16:31 ttyp2
crw-r--r--   1 root     root       4, 195 Sep  8 16:31 ttyp3
crw-r--r--   1 root     root       4, 196 Sep  8 16:31 ttyp4
crw-r--r--   1 root     root       4, 197 Sep  8 16:31 ttyp5
crw-r--r--   1 root     root       4, 198 Sep  8 16:31 ttyp6
crw-r--r--   1 root     root       4, 199 Sep  8 16:31 ttyp7
crw-r--r--   1 root     root       4, 200 Sep  8 16:31 ttyp8
crw-r--r--   1 root     root       4, 201 Sep  8 16:31 ttyp9
crw-r--r--   1 root     root       4, 202 Sep  8 16:31 ttypa
crw-r--r--   1 root     root       4, 203 Sep  8 16:31 ttypb
crw-r--r--   1 root     root       4, 204 Sep  8 16:31 ttypc
crw-r--r--   1 root     root       4, 205 Sep  8 16:31 ttypd
crw-r--r--   1 root     root       4, 206 Sep  8 16:31 ttype
crw-r--r--   1 root     root       4, 207 Sep  8 16:31 ttypf
-rw-------   1 root     root        63488 Sep  8 16:31 ttys0
crw-r--r--   1 root     root       4,  67 Sep  8 16:31 ttys3
crw-r--r--   1 root     root       1,   5 Sep  8 16:31 zero

boot_disk/etc:
total 173
-rw-r--r--   1 root     root           53 Sep  8 18:48 boot.env
-rwxr-xr-x   1 root     root        27408 Sep  8 18:48 e2fsck*
-rwxr-xr-x   1 root     root        18540 Sep  8 18:48 fdisk*
-rw-r--r--   1 root     root           69 Oct  8 12:27 fstab
-r-x------   1 root     root        13312 Sep  8 18:48 getty*
-rw-r--r--   1 root     root          334 Sep  8 18:48 group
-rw-r--r--   1 root     root           12 Sep  8 18:48 host.conf
-rw-r--r--   1 root     root           62 Sep  8 18:48 hosts
-r-x------   1 root     root         6684 Sep  8 18:48 ifconfig*
-rwxr-xr-x   1 root     root        11492 Sep  8 18:48 init*
-rw-r--r--   1 root     root         1017 Sep  9 22:12 inittab
-rw-r--r--   1 root     root            0 Oct  8 12:19 issue
-rw-r-----   1 root     root         5137 Sep  8 18:48 login.defs
```

```
-rwxr-xr-x   1 root     root        14028 Sep  8 18:48 mke2fs*
-rwxr-x---   1 root     root         2436 Sep  8 18:48 mkswap*
-rwxr-xr-x   1 root     root        11288 Sep  8 18:48 mount*
-rw-r--r--   1 root     root          327 Sep  8 18:48 passwd
-rwxr-xr-x   1 root     root          383 Sep 10 16:02 profile*
-rw-r--r--   1 root     root           94 Sep  8 18:48 protocols
-rwxr-xr-x   1 root     root          334 Oct  8 12:27 rc*
-rwxr-xr-x   1 root     root         9220 Sep  8 18:48 reboot*
-r-x------   1 root     root         4092 Sep  8 18:48 route*
-rw-r--r--   1 root     root           20 Sep  8 18:48 securetty
-rw-r--r--   1 root     root         9749 Sep  8 18:48 services
-rw-r--r--   1 root     root           36 Sep  8 18:48 shells
-rwxr-xr-x   1 root     root        13316 Sep  8 18:48 shutdown*
-rwxr-xr-x   1 root     root         2496 Sep  8 18:48 swapoff*
-rwxr-xr-x   1 root     root         2496 Sep  8 18:48 swapon*
-rw-r--r--   1 root     root         5314 Sep  8 18:48 termcap
-rwxr-xr-x   1 root     root         5412 Sep  8 18:48 umount*
-rw-r--r--   1 root     root          224 Sep  8 18:48 utmp
-rw-r--r--   1 root     root          280 Sep  8 18:48 wtmp

boot_disk/lib:
total 629
-rwxr-xr-x   1 root     root        17412 Sep 10 14:58 ld.so*
-rwxr-xr-x   1 root     root       623620 Sep  8 18:33 libc.so.4*

<sect2>Utility Disk ls-lR Directory Listing
<p>
The utility listing is of directory util_disk:

total 1
drwxr-xr-x   2 root     root         1024 Sep 10 16:05 bin/

util_disk/bin:
total 897
-rwxr-xr-x   1 root     root        41984 Sep 10 14:11 cpio*
-rwxr-xr-x   1 root     root       504451 Sep  9 21:39 ftape.o*
-rwxr-xr-x   1 root     root        63874 Sep  9 21:40 gzip*
-rwxr-xr-x   1 root     root        13316 Sep  9 21:34 insmod*
-rwxr-xr-x   1 root     root           58 Sep  9 21:34 lsmod*
-rwxr-xr-x   1 root     root         3288 Sep  9 21:34 mknod*
-rwxr-xr-x   1 root     root         9220 Sep  9 21:34 rmmod*
-rwxr-xr-x   1 root     root       226308 Sep  9 22:13 tar*
```

4.2 Shell Scripts to Build Diskettes

There are two shell scripts:

- mkroot - builds a root or boot/root diskette.
- mkutil - builds a utility diskette.

Both are currently configured to run in the parent directory of boot_disk and util_disk, each of which contains everything to be copied to it's diskette. Note that these shell scripts will *NOT* automatically set up and copy all the files for you - you work out which files are needed, set up the directories and copy the files to those directories. The shell scripts are samples which will copy the contents of those directories. Note that they are primitive shell scripts and are not meant for the novice user.

The scripts both contain configuration variables at the start which allow them to be easily configured to run anywhere. First, set up the model directories and copy all the required files into them. Then check the configuration variables in the shell scripts and change them as required before running the scripts.

4.2.1 mkroot - Make Root or Boot/Root Diskette

```
# mkroot: make a boot/boot disk - creates a boot/root diskette
#       by building a file system on it, then mounting it and
#       copying required files from a model.
#       Note: the model to copy from from must dirst be set up,
#       then change the configuration variables below to suit
#       your system.
#
# usage: mkroot [nokernel]
#       if the parameter is omitted, then the kernel and LILO
#       are copied.

# Copyright (c) Graham Chapman 1994. All rights reserved.
# Permission is granted for this material to be freely
# used and distributed, provided the source is acknowledged.
# No warranty of any kind is provided. You use this material
# at your own risk.

# Configuration variables...
BOOTDISKDIR=./boot_disk      # name of boot disk directory
MOUNTPOINT=./mnt             # temporary mount point for diskette
LILODIR=/sbin                # directory containing lilo
LILOBOOT=/boot/boot.b        # lilo boot sector
LILOMSG=./lilo.msg           # lilo message to display at boot time
LILOCONFIG=./lilo.conf       # lilo parms for boot/root diskette
DISKETTEDEV=/dev/fd0         # device name of diskette drive

echo $0: create boot/root diskette
echo Warning: data on diskette will be overwritten!
echo Insert diskette in $DISKETTEDEV and and press any key...
read anything

mke2fs $DISKETTEDEV
```

```
if [ $? -ne 0 ]
then
        echo mke2fs failed
        exit
fi

mount -t ext2 $DISKETTEDEV $MOUNTPOINT
if [ $? -ne 0 ]
then
        echo mount failed
        exit
fi

# copy the directories containing files
for i in bin etc lib
do
        cp -dpr $BOOTDISKDIR/$i $MOUNTPOINT
done

# copy dev *without* trying to copy the files in it
cp -dpR $BOOTDISKDIR/dev $MOUNTPOINT

# create empty directories required
mkdir $MOUNTPOINT/proc
mkdir $MOUNTPOINT/tmp
mkdir $MOUNTPOINT/mnt
mkdir $MOUNTPOINT/usr

# copy the kernel
if [ "$1" != "nokernel" ]
then
        echo "Copying kernel"
        cp $BOOTDISKDIR/vmlinux $MOUNTPOINT
        echo kernel copied

        # setup lilo
        cp $LILOBOOT $MOUNTPOINT
        cp $LILOMSG $MOUNTPOINT
        $LILODIR/lilo -C $LILOCONFIG
        echo LILO installed
fi

umount $MOUNTPOINT

echo Root diskette complete
```

4.2.2 mkutil - Make Utility Diskette

```
# mkutil: make a utility diskette - creates a utility diskette
#       by building a file system on it, then mounting it and
#       copying required files from a model.
#       Note: the model to copy from from must first be set up,
#       then change the configuration variables below to suit
#       your system.

# Copyright (c) Graham Chapman 1994. All rights reserved.
# Permission is granted for this material to be freely
# used and distributed, provided the source is acknowledged.
# No warranty of any kind is provided. You use this material
# at your own risk.

# Configuration variables...
UTILDISKDIR=./util_disk      # name of directory containing model
MOUNTPOINT=./mnt             # temporary mount point for diskette
DISKETTEDEV=/dev/fd0         # device name of diskette drive

echo $0: create utility diskette
echo Warning: data on diskette will be overwritten!
echo Insert diskette in $DISKETTEDEV and and press any key...
read anything

mke2fs $DISKETTEDEV
if [ $? -ne 0 ]
then
        echo mke2fs failed
        exit
fi

# Any file system type would do here
mount -t ext2 $DISKETTEDEV $MOUNTPOINT
if [ $? -ne 0 ]
then
        echo mount failed
        exit
fi

# copy the directories containing files
cp -dpr $UTILDISKDIR/bin $MOUNTPOINT

umount $MOUNTPOINT

echo Utility diskette complete
```

5 FAQ

5.1 Q. How can I make a boot disk with a XXX driver?

The easiest way is to obtain a Slackware kernel from your nearest Slackware mirror site. Slackware kernels are generic kernels which atttempt to include drivers for as many devices as possible, so if you have a SCSI or IDE controller, chances are that a driver for it is included in the Slackware kernel.

Go to the a1 directory and select either IDE or SCSI kernel depending on the type of controller you have. Check the xxxxkern.cfg file for the selected kernel to see the drivers which have been included in that kernel. If the device you want is in that list, then the corresponding kernel should boot your computer. Download the xxxxkern.tgz file and copy it to your boot diskette as described above in the section on making boot disks.

You must then check the root device in the kernel, using the rdev command:

```
rdev vmlinuz
```

Rdev will then display the current root device in the kernel. If this is not the same as the root device you want, then use rdev to change it. For example, the kernel I tried was set to /dev/sda2, but my root scsi partition is /dev/sda8. To use a root diskette, you would have to use the command:

```
rdev vmlinuz /dev/fd0
```

If you want to know how to set up a Slackware root disk as well, that's outside the scope of this HOWTO, so I suggest you check the Linux Install Guide or get the Slackware distribution. See the section in this HOWTO titled "References".

5.2 Q. How do I update my boot floppy with a new kernel?

Just copy the kernel to your boot diskette using the dd command for a boot diskette without a filesystem, or the cp command for a boot/root disk. Refer to the section in this HOWTO titled "Boot" for details on creating a boot disk. The description applies equally to updating a kernel on a boot disk.

5.3 Q. How do I remove LILO so that I can use DOS to boot again?

This is not really a Bootdisk topic, but it is asked so often, so: the answer is, use the DOS command:

```
FDISK /MBR
```

MBR stands for Master Boot Record, and it replaces the boot sector with a clean DOS one, without affecting the partition table. Some purists disagree with this, but even the author of LILO, Werner Almesberger, suggests it. It is easy, and it works.

You can also use the dd command to copy the backup saved by LILO to the boot sector - refer to the LILO documentation if you wish to do this.

5.4 Q. How can I boot if I've lost my kernel AND my boot disk?

If you don't have a boot disk standing by, then probably the easiest method is to obtain a Slackware kernel for your disk controller type (IDE or SCSI) as described above for "How do I make a boot disk with a XXX driver?". You can then boot your computer using this kernel, then repair whatever damage there is.

The kernel you get may not have the root device set to the disk type and partition you want. For example, Slackware's generic scsi kernel has the root device set to /dev/sda2, whereas my root Linux partition happens to be /dev/sda8. In this case the root device in the kernel will have to be changed.

You can still change the root device and ramdisk settings in the kernel even if all you have is a kernel, and some other operating system, such as DOS.

Rdev changes kernel settings by changing the values at fixed offsets in the kernel file, so you can do the same if you have a hex editor available on whatever systems you do still have running - for example, Norton Utilities Disk Editor under DOS. You then need to check and if necessary change the values in the kernel at the following offsets:

```
0x01F8  Low byte of RAMDISK size
0x01F9  High byte of RAMDISK size
0x01FC  Minor device number - see below
0X01FD  Major device number - see below
```

The ramdisk size is the number of blocks of ramdisk to create. If you want to boot from a root diskette then set this to decimal 1440, which is 0x05A0, thus set offset 0x01F8 to 0xA0 and offset 0x01F9 to 0x05. This will allocate enough space for a 1.4Mb diskette.

The major and minor device numbers must be set to the device you want to mount your root filesystem on. Some useful values to select from are:

```
device          major minor
/dev/fd0            2     0    1st floppy drive
/dev/hda1           3     1    partition 1 on 1st IDE drive
/dev/sda1           8     1    partition 1 on 1st scsi drive
/dev/sda8           8     8    partition 8 on 1st scsi drive
```

Once you have set these values then you can write the file to a diskette using either Norton Utilities Disk Editor, or a program called rawrite.exe. This program is included in several distributions, including the SLS and Slackware distributions. It is a DOS program which writes a file to the "raw" disk, starting at the boot sector, instead of writing it to the file system. If you use Norton Utilities, then you must write the file to a physical disk starting at the beginning of the disk.

5.5 Q. How can I make extra copies of boot/root diskettes?

It is never desirable to have just one set of rescue disks - 2 or 3 should be kept in case one is unreadable.

The easiest way of making copies of any diskettes, including bootable and utility diskettes, is to use the dd command to copy the contents of the original diskette to a file on your hard drive, and then use the same command to copy the file back to a new diskette. Note that you do not need to, and should not, mount the diskettes, because dd uses the raw device interface.

To copy the original, enter the command:

```
dd if={$<$}device{$>$} of={$<$}filename{$>$}
where   {$<$}device{$>$} = the device name of the diskette
        drive
and     {$<$}filename{$>$} = the name of the file where you
        want to copy to
```

For example, to copy from /dev/fd0 to a temporary file called /tmp/diskette.copy, I would enter the command:

```
dd if=/dev/fd0 of=/tmp/diskette.copy
```

Omitting the "count" parameter, as we have done here, means that the whole diskette of 2880 (for a high-density) blocks will be copied.

To copy the resulting file back to a new diskette, insert the new diskette and enter the reverse command:

```
dd if=<filename> of=<device>
```

Note that the above discussion assumes that you have only one diskette drive. If you have two of the same type, then you can copy diskettes using a command like:

```
dd if=/dev/fd0 of=/dev/fd1
```

5.6 Q. How can I boot without typing in "ahaxxxx=nn,nn,nn" every time?

Where a disk device cannot be autodetected it is necessary to supply the kernel with a command device parameter string, such as:

```
aha152x=0x340,11,3,1
```

This parameter string can be supplied in several ways using LILO:

- By entering it on the command line every time the system is booted via LILO. This is boring, though.
- By using the LILO "lock" keyword to make it store the command line as the default command line, so that LILO will use the same options every time it boots.
- By using the APPEND statement in the lilo config file. Note that the parameter string must be enclosed in quotes.

For example, a sample command line using the above parameter string would be:

```
vmlinux aha152x=0x340,11,3,1 root=/dev/sda1 lock
```

This would pass the device parameter string through, and also ask the kernel to set the root device to /dev/sda1 and save the whole command line and reuse it for all future boots.

A sample APPEND statement is:

```
APPEND = "aha152x=0x340,11,3,1"
```

Note that the parameter string must NOT be enclosed in quotes on the command line, but it MUST be enclosed in quotes in the APPEND statement.

Note also that for the parameter string to be acted on, the kernel must contain the driver for that disk type. If it does not, then there is nothing listening for the parameter string, and you will have to rebuild the kernel to include the required driver. For details on rebuilding the kernel, cd to /usr/src/linux and read the README, and read the Linux FAQ and Installation HOWTO. Alternatively you could obtain a generic kernel for the disk type and install that.

Readers are strongly urged to read the LILO documentation before experimenting with LILO installation. Incautious use of the "BOOT" statement can damage partitions.

6 References

In this section, vvv is used in package names in place of the version, to avoid referring here to specific versions. When retrieving a package, always get the latest version unless you have good reasons for not doing so.

6.1 LILO - Linux Loader

Written by Werner Almesberger. Excellent boot loader, and the documentation includes information on the boot sector contents and the early stages of the boot process.

Ftp from: `tsx-11.mit.edu:/pub/linux/packages/lilo/lilo.vvv.tar.gz` also on sunsite and mirror sites.

6.2 Linux FAQ and HOWTOs

These are available from many sources. Look at the usenet newsgroups `news.answers` and `comp.os.linux.announce`.

Ftp from: `sunsite.unc.edu:/pub/Linux/docs`

- FAQ is in `/pub/linux/docs/faqs/linux-faq`
- HOWTOs are in `/pub/Linux/docs/HOWTO`

For WWW, start at the Linux documentation home page:

```
http://sunsite.unc.edu/mdw/linux.html
```

If desperate, send mail to:

```
mail-server@rtfm.mit.edu
```

with the word "help" in the message, then follow the mailed instructions.

Note: if you haven't read the Linux FAQ and related documents such as the Linux Installation HOWTO and the Linux Install Guide, then you should not be trying to build boot diskettes.

6.3 Rescue Shell Scripts

Written by Thomas Heiling. This contains shell scripts to produce boot and boot/root diskettes. It has some dependencies on specific versions of other software such as LILO, and so might need some effort to convert to your system, but it might be useful as a starting point if you wanted more comprehensive shell scripts than are provided in this document.

Ftp from: `sunsite.unc.edu:/pub/Linux/system/Recovery/rescue.tgz`

6.4 SAR - Search and Rescue

Written by Karel Kubat. SAR produces a rescue diskette, using several techniques to minimize the space required on the diskette. The manual includes a description of the Linux boot/login process.

Ftp from: `ftp.icce.rug.nl:/pub/unix/SAR-vvv.tar.gz`

The manual is available via WWW from:

`http://www.icce.rug.nl/karel/programs/SAR.html`

6.5 Slackware Distribution

Apart from being one of the more popular Linux distributions around, it is also a good place to get a generic kernel. It is available from almost everywhere, so there is little point in putting addresses here.