MCC Interim Linux

The MCC Distribution of the Linux Operating System

Version 0.99.p8+

May 1993

Published by Manchester Computing Centre, University of Manchester.

1. Introduction

The MCC interim versions of Linux are designed to allow people who are not Unix experts to install a version of the Linux operating system on a PC. The installed system should be self-contained, but easy to extend.

1.1. What is Linux?

Linux is an operating system written by the Finnish programmer Linus B Torvalds. It is designed to look like the Unix operating system, and the source of the entire system is available. Binaries and source can be distributed freely under the terms of the GNU Public License. See Section 1.3 [Copyright and conditions of distribution], page 4. Programs from the various System V and BSD versions of Unix should compile and run under Linux with few changes. Linux aims to conform as closely as it can to the various POSIX, ISO, and ANSI standards.

Linux runs only on machines which have processors compatible with the Intel 386 or 486. If you intend to run Linux, your computer must have appropriate hardware and sufficient resources:

- You must have at least 2 Mb of memory to run Linux. For good performance, you need at least 4 Mb of memory, and the X window system does not work well without at least 8 Mb.
- Linux requires an ISA or EISA bus. At present it does not support the MCA bus architecture. This means that is does not run on a PS2 machine.
- Linux supports generic AT drives, including IDE and 16 bit hard disk controllers with MFM or RLL. It supports generic XT controllers (8 bit hard disk controllers with MFM or RLL). It supports SCSI disks and CD ROMs, provided the SCSI adaptor is supported. Supported adaptors include Adaptec 1542 (but not 1522), Adaptec 1740 in extended mode (not in 1542 compatible mode), Seagate ST-01 and ST-02, Future Domain TMC88x series, any board based on the TMC950 chip, TMC1660 and 1680, Ultrastor 14F, 24F, and 34F, and WD7000.
- You will need disk space for swapping (or paging) and for file systems. A small installation can share a 40 Mb disk with DOS, but a full installation including the X window system may require a full 40 Mb for Linux files alone. See Chapter 3 [Preparing to Install Linux], page 11.
- Linux supports several Ethernet cards, but the 'official' kernel currently contains code only for Western Digital 8003 and 8013 cards. Patches for other cards are available, and will eventually be included in the 'official' kernel. There is not at present a patch to support the RM Ethernet cards, which are the most common cards in Manchester, but we hope to modify the drivers to support these cards in the near future.
- Text mode supports VGA, EGA, CGA, and Hercules cards. Linux supports graphics and X11 with EGA, normal VGA, some SVGA cards (ET3000, ET4000, Paradise, and some Trident chips), some S3 cards (but not Diamond Stealth), 8514/A, and Hercules.

Detailed information about currently supported hardware is published in the Linux hardware list, which is available by anonymous ftp. See Section 2.1 [Getting MCC interim Linux by anonymous ftp], page 7.

Among the features of Linux are the following:

- It is a multi-tasking system: several programs can run at the same time.
- It is a multi-user system: several users can use the same machine at the same time.
- Linux runs in 386 protected mode. Different processes are isolated from each other, so that one program cannot bring the system down.
- Executable files are demand-loaded, so that parts of a program are in memory only when those parts are being executed.
- Compiled programs are linked with dynamically loaded shared libraries to reduce disk and memory usage.
- Executables share memory pages until one of them writes to it. This increases speed and decreases memory use.
- Programs access virtual memory which is paged to disk. If you allocate up to 16 swap areas, each up to 16 Mb in size, all of this memory is available to running processes in a shared pool.
- Physical memory is organised into a unified memory pool or cache, which changes in size depending on the system load.
- The kernel can emulate a maths coprocessor if you have none. If you have a coprocessor, you can recompile the kernel without the emulation code to save memory.
- The kernel supports multiple virtual consoles, which allow several independent login sessions on the system console.
- The kernel supports several file systems, including its own extended and extended version 2 file systems, MINIX, XENIX, and DOS compatible file systems, and a special *proc* file system for getting information about running processes. Linux cannot access DOS 6 compressed partitions.
- Linux supports TCP/IP networking, including ftp, telnet, NFS, and other protocols.

The software which is distributed in the MCC interim versions of Linux comes from a variety of sources. Much of it comes from the GNU project, some of it is taken from the published BSD sources, and some has been written specifically for Linux.

WARNING:

> Linux is an operating system which is very different from DOS. It requires a certain minimal amount of attention if it is to work well. In particular, you must **never** simply turn off a machine on which Linux is running. For more information about using Linux safely, see Section 7.1 [Caring for your Linux system], page 35.

1.2. What are the MCC interim versions of Linux?

When Linux version 0.10 first appeared in the autumn of 1991, it was released by Linus Torvalds as a pair of floppy disks, known as the *boot* and *root* disks. Linus himself released several utilities which did not fit on these disks. Eventually quite a number of contributions were available. Software distribution was rather chaotic, and you needed a lot of effort to collect a complete base system.

The Manchester Computing Centre (MCC) is a part of the University of Manchester, supporting computing at the University, at UMIST (the University of Manchester Institute of Science and Technology), and at several other British universities.

We began to make Linux available by anonymous ftp in November 1991, and we released the first MCC interim version of Linux (version 0.12+) the following February. This version made use of ramdisk code written by Theodore Ts'o to mount a virtual root disk at boot time from the kernel floppy. (This type of distribution was used earlier by Andrew Tannenbaum to distribute his operating system MINIX.) With version 0.99p8+, we abandoned the ramdisk-as-root and adopted a combined bootable root disk. The first combined root disk was created by H J Lu (who made essential use of Werner Almesberger's LILO software). Unlike H J's bootable root disk, the MCC disk does not contain an executable shell.

Very shortly after the first MCC interim version of Linux appeared, other people released similar versions: Dave Safford's TAMU releases and Martin Junius's MJ versions were eventually followed by Peter Macdonald's massive, comprehensive SLS releases and H J Lu's small base systems. What distinguishes the MCC versions from these others?

From its first release, MCC interim Linux has included basic utilities, the GNU C compiler, and the sources to the current kernel release. The software included in MCC distributions comes from a wide variety of sources, but we try to make sources to all of it available, though not as part of the distribution. MCC interim versions aim to provide a stable base system, which can be installed in a short time, and to which other software can be added with relatively little effort. Every binary file in an MCC distribution has been compiled under that version of the kernel, by that version of `gcc`, is linked with that version of the libraries, and has been tested to some extent. The only exception to this is the GNU C compiler and libraries, which we take directly from H J Lu's distribution.

We install this version of Linux on small 386-SX machines for our C and Unix courses, and we need to be able to install it on twelve machines in about one hour. We also distribute this version of Linux to our clients in Manchester. We make no attempt to include a wide variety of packages, or large packages such as X386. The *emacs* and *info* packages were added to MCC interim Linux only because we have been using them for our courses. People who want a distribution which contains nearly everything should get the SLS versions, which are about five times the size of MCC, and quite comprehensive.

Our versions are called *interim* because they are not intended to be final or official. They are small, harmonious, and moderately tested. They do not conform to everyone's taste — what release does? — but they should provide a stable base to which other software can be added.

This version of Linux is prepared to allow you to upgrade an existing system, as well as to install a complete new system. The limitations are mainly that partitions must contain *minix* or *ext2* file systems: *ext* and *xiafs* file systems are not supported in the installation, though they can be mounted from the installed system with no difficulty. Upgrading an old system looks just like installing a new system, except that you don't need to run '`mkswap`' and '`mkfs`' before installing. Instead you should run '`fsck`', which you can do from the boot disk. Old binaries are deleted, except for some old libraries, which may be required by programs which you have acquired from somewhere else.

Text files, especially the configuration files in /etc, are more of a problem. We don't want to throw away the old files, which you may have spent some considerable time polishing, but we must install new configuration files, or else the system may not boot properly, or may work badly if it boots at all. For this reason, the installation creates a special directory named /backupdirs. In this directory it creates subdirectories such as etc, etc/lilo, root, and home/user. Any old configuration files which might be overwritten during the upgrade are moved to these directories before the new files are installed. After installation, you may cautiously compare the new files with the old ones, and incorporate your personal preferences into the new files.

You must be particularly careful with files such as /etc/inittab or /etc/rc, since mistakes in these may prevent the system from booting, or cause serious problems while the system is running. It is a good idea to read the man pages for init and inittab before making any changes to these files. A number of other rc files are created in /etc during the installation; these are described to some extent in /etc/inittab. See Chapter 6 [Tailoring MCC Interim Linux to Your Taste], page 31.

1.3. Copyright and conditions of distribution

Software today comes under a wide variety of copyright restrictions, and it is important to know what conditions apply to the software you use. The software in MCC interim versions of Linux, and other software which can be added to it, is copyrighted in a variety of ways. In general, all of it is available to be used by anyone, and the source to all of it is available as well. If you intend to use Linux or any of its parts for commercial purposes, you should get the source files and read the copyrights contained in them. (A list of the sources can be found in Appendix A [Acknowledgments], page 51.) Most Linux software falls under one of the following categories:

GNU *copyleft*

: The GNU General Public License applies to the Linux kernel, to the GNU C and C++ compilers, to all utilities distributed by the Free Software Foundation, and to many contributed utilities. This copyright, also known as a *copyleft*, is designed to ensure that the source to software is available, and that you can give away or sell copies of the source and of compiled binaries. There is no warranty, but you are obliged to notify anyone of any changes which you have made to the original. You are obliged to publish on each copy an appropriate copyright notice and a disclaimer of warranty. Works derived from copylefted works must be released under the same terms. The terms are specified in the GNU General Public License, which is available by anonymous ftp. See Section 2.1 [Getting MCC interim Linux by anonymous ftp], page 7.

GNU library license

: The GNU library license applies to the GNU C++ library and to all works that must be linked with this library. This allows commercial binaries to be supplied in an unlinked form, so that purchasers can link the binaries with other versions of the library.

BSD copyright

: The BSD copyright applies to all source from the University of California at Berkeley. Source may be reused freely, but it should contain a notice to the effect that all or part of the software was developed by the University of California at Berkeley. Moreover, the name of the University may not be used to endorse or promote such products.

MIT copyright
: The X Window System, version 11, is copyrighted by the Massachusetts Institute of Technology, and 'X Window System' is a trademark owned by MIT. Most of the source for this software was developed by the MIT X Consortium, and is covered by the MIT copyright. This is similar to the BSD copyright in permitting free commercial use of MIT code, subject to the inclusion of similar notices giving credit for any MIT code used in the final product.

Public domain software
: Some programs in this distribution are in the public domain. They have no copyright attached and can be used in any way without any obligation or restriction.

1.4. Bugs and warnings

The MCC interim release of Linux assembles code from many sources. Bugs have been fixed in some cases, but created in others. I would appreciate any bug reports or fixes. Please send them by e-mail to me at `LeBlanc@mcc.ac.uk`.

1. The command '`mkfs.ext2`' sometimes fails to create a file system because there is not enough space for the last superblock. To work around this, reduce the number of blocks slightly. Remy Card has promised a fix eventually.
2. The command '`umount -a -t noproc`' appears to leave the *proc* file system mounted, but removes its entry from `/etc/mtab`.
3. Certain commands when received by the ftp daemon cause the response '`425 Can't build data connection: Transport endpoint is already connected`'. If you are lucky, try them again, and they will work.
4. The '`rcp`' command sometimes hangs without copying the last partially full buffer.
5. The command '`makewhatis -c`' puts a lot of rubbish in `/usr/man/whatis`.
6. Installation failed on systems with SCSI disks with more than 1024 cylinders. Fixed 6 May 1993.
7. The file `rc.net` created by install.net from the *tcpip* package created a useless file in `/` during every boot. Fixed 6 May 1993.

2. Getting a Copy of MCC Interim Linux

There are two ways at present by which you can get a copy of MCC interim versions of Linux: by anonymous ftp, and by visiting MCC. The latter method is available as a service only for students and members of staff of the University of Manchester and of UMIST.

2.1. Getting MCC interim Linux by anonymous ftp

MCC interim versions of Linux are released on `ftp.mcc.ac.uk [130.88.203.12]` in the directory `/pub/linux/mcc-interim`. In this directory there are subdirectories `dos-utils`, which contains some commands which you may need if you install Linux from DOS, `0.99.p8+`, which contains the current release, and `old`, which contains one or more older releases. The current MCC release is also available from `sunsite.unc.edu [152.2.22.81]` in the directory `/pub/Linux/distributions/MCC`, and also from `nic.funet.fi [128.214.6.100]` in `/pub/OS/Linux/images/MCC-interim`. Other sites sometimes maintain mirrors of the current MCC distribution.

The directory `0.99.p8+` contains the current `README` files and the file `BUGS+WARNINGS`, which contain important information. (The contents of all of these files have been incorporated into the present document.) It also contains the following subdirectories:

`images` This contains the gzipped images of the bootable root disks, `boot-uk.z` (for UK keyboards) and `boot-us.z` (for US keyboards). '`gzip`' is the GNU compression utility.

`packages` This contains the packages which are optional parts of the MCC interim distribution.

`kernels` This contains additional kernels which may be installed as if they were packages. You should not install more than one of these kernels. They contain no support for SCSI or XT disks, and two of them contain no support for NFS or networking. All six MCC kernels contain maths coprocessor emulation code, and support for IDE hard disks and for the *minix*, *ext2*, *msdos*, and *proc* file systems.

- *boot-uk* SCSI, XT, NFS, networking, UK keyboard.
- *boot-us* SCSI, XT, NFS, networking, US keyboard.
- *ide-uk* No SCSI, no XT, no NFS, no networking, UK keyboard.
- *ide-us* No SCSI, no XT, no NFS, no networking, US keyboard.
- *ipide-uk* No SCSI, no XT, UK keyboard with networking and NFS support.
- *ipide-us* No SCSI, no XT, US keyboard with networking and NFS support.

`source-files`

This contains the source files which were used to compile all binaries in the current distribution. It also contains the file `manpages.tar.z`

After you decide what you need, you may transfer those files in binary mode to any machine which supports ftp. The bootable root disk must be gunzipped and copied to a high density floppy disk, either 3.5 inch 1.44 Mb or 5.25 inch 1.2 Mb, of the sort which can be used to boot the PC on which you intend to install or upgrade Linux. If you have a Linux system which you are upgrading, you may put a formatted floppy into your 0 drive (which DOS calls '`A:`') and give these commands:

```
fdformat /dev/fd0H1440                 (for 3.5 inch disks)
fdformat /dev/fd0h1200                 (for 5.25 inch disks)
gunzip boot-uk.z
dd if=boot-uk of=/dev/fd0 bs=18432     (for 3.5 inch disks)
dd if=boot-uk of=/dev/fd0 bs=15360     (for 5.25 inch disks)
```

If you are installing Linux on a machine which already contains a version of Linux, you may wish to put the MCC packages in a directory in the existing file system. MCC interim versions at present support only the *minix* and *ext2* file systems. The kernel will not allow you to mount older *ext* or *xiafs* file systems. The packages you wish to install should be in a single directory, and they must have the same file names (in lower case) as they do in the MCC distribution directory. You may put all the packages in a DOS directory on the same machine, or you may copy them to floppies. The MCC interim distribution prefers DOS-format floppies, though *minix* or *ext2* floppies will work as well. DOS format floppies will hold more, and will not produce spurious error messages during installation.

If you are installing from DOS, you need commands which will gunzip the disk image and copy it to a disk. The commands '`GZIP.EXE`', '`RAWRITE.EXE`', and '`RAWRITE3.COM`' can be found at `ftp.mcc.ac.uk` in the directory `/pub/linux/mcc-interim/dos-utils`. Transfer '`GZIP.EXE`' and one of the '`RAWRITE`' commands in binary mode. The command '`GZIP.EXE`' is a self-unpacking archive, which creates a binary of the same name and a documentation file. Documentation for the two '`RAWRITE`' commands can be found in the `dos-utils` directory with the commands; you should normally transfer documentation files in text mode. After unpacking '`GZIP.EXE`', you should give the following commands:

```
FORMAT A:
GZIP -D BOOT-UK.Z
RAWRITE
Enter source file name: BOOT-UK
Enter destination drive: A:
Please insert a formatted diskette into drive A: and press -ENTER- :
```

The program '`RAWRITE3`' is screen-oriented, and may work rather faster than the older '`RAWRITE`'.

If you are installing Linux on a system which already contains a DOS partition, you may find it convenient to put MCC packages in a single directory on that partition. You can also copy them to DOS format floppies. As explained above, the file names must be the same as in the `packages` directory, though under DOS they will, of course, be in upper-case. If you get all packages, they will fit on floppies as follows:

3.5 inch high density floppies:

disk 1	*gcc*, *tcpip*
disk 2	*emacs*, *info*

disk 3	*bison*, *gpp*, *groff*
disk 4	*base*, *flex*, *gprof*, *linux*, *tar*
disk 5	*gawk*, *gdb*, *kermit*, *patches*

5.25 inch high density floppies:

disk 1	*gcc*
disk 2	*flex*, *gdb*, *info*
disk 3	*gpp*, *gprof*, *tcpip*
disk 4	*bison*, *emacs*, *gawk*, *kermit*, *patches*, *tar*
disk 5	*base*, *linux*
disk 6	*groff*

In either case, the final disk contains enough space for one of the extra kernels, or for anything else you wish. Any gzipped tar file with the extension '`.tgz`' which is placed on these disks can be installed. It is unzipped and untarred in the root directory. If it contains files `install.setup` and `install.clean`, these are executed respectively before and after untarring the archive. If you wish, you may include a file `install.info`, which is copied to the screen during installation. The MCC packages all contain a file *filename*`.crc`, which must have the same basename as the '`.tgz`' file, and which has been generated with the command '`brik -Gb`'. If such a file exists, the installation uses it to verify the included checksums after `install.clean` is executed.

You may omit any packages you wish. Obviously, you cannot recompile the kernel unless you also install *gcc*. *Bison*, *flex*, *gdb*, *gpp*, and *gprof* are also of little use unless *gcc* is installed.

The default kernels in the UK and US boot disks contain all SCSI disk drivers and XT disk drivers in the kernel as distributed by Linus Torvalds, as well as networking, NFS, and kernel maths coprocessor emulation.

It is also possible to install MCC interim packages from a directory on a remote machine which can be mounted on your Linux system using NFS. To do this, you must have a supported Western Digital Ethernet card. You must already have networking software installed and configured. If necessary, you can install the *base* and *tcpip* packages from a floppy disk, get the networking software working, and then install other packages.

2.2. Getting MCC interim Linux from the MCC shop

MCC interim Linux is available from the MCC shop to members of the University of Manchester and of UMIST. You can buy a set of diskettes containing the UK bootable root disk, all the packages, and the two optional kernels with UK keyboards and with no SCSI support. See Section 2.1 [Getting MCC interim Linux by anonymous ftp], page 7, where these optional kernels are described. You can also obtain a free copy of this document.

3. Preparing to Install Linux

There are several things which you must do before installing MCC interim Linux besides getting the appropriate files and making a bootable root disk. The target machine must have free space which is not included in already allocated partitions. In some cases this means that you must back up your hard disk, create a smaller DOS partition, using the DOS command '`FDISK`', and restore your DOS files to the new DOS partition.

There are procedures which allow you to change the size of a DOS partition without backing up your files, but we do not recommend such procedures. For more information, see Section 7.2.2 [Dividing up your disk], page 37.

3.1. Disk space required for Linux files

You may wish to have a small Linux system with only a few of the MCC interim packages. The following table gives the approximate size required for each package. Remember that each file system also uses up a few hundred Kb on each partition, depending on the type of file system. Extended(2) file systems also reserve a percentage of each partition for the superuser; the exact percentage can be selected by an option to '`mkfs`'. (The MCC distribution does not allow you to change the 5% default before installation.) The package *Base* must be installed from the boot floppy before any other packages.

Package	Size in Kb	Contents
Base	1841	basic binaries and text (on boot floppy)
base	1241	basic binaries, texts, and man pages
bison	100	the GNU yacc-compatible parser generator
emacs	2040	the emacs editor, without info files
flex	204	the GNU fast lexical analyser generator
gawk	141	the GNU version of awk
gcc	3375	the GNU C compiler (without g++)
gdb	767	the GNU debugger
gpp	2222	the GNU C++ compiler g++
gprof	513	the GNU profiler for use with gcc and g++
groff	2250	the GNU clone of the nroff/troff text formatter
info	2646	the GNU utility info, plus selected info files
kermit	352	the Kermit communications utility
linux	2385	the kernel source for Linux 0.99 patch level 8
patches	251	the patches used to compile this release
tar	72	the GNU version of tar
tcpip	658	commands and daemons for networking
Total	21058	

The amount of disk space you require depends on what software you intend to install. For all MCC packages, you need about 22 Mb. For the X window system (which is not included in the

MCC distribution), you need a further 17 Mb. TEX (which is also not included in the MCC distribution), requires at least 10 Mb more. The source files for all MCC packages (which are also not part of the distribution) require a further 25 Mb. Additionally, you must consider the space you need for work, perhaps an additional 5 or 10 Mb, or 100 Mb or more for large projects.

On the other hand, you may plan to have a large system. In that case, you may wish to distribute your files over several partitions. Linux allows you to have up to 16 partitions on a SCSI disk, and up to 64 partitions on an IDE disk. How you divide your disk is up to you, of course, but there are some points you may wish to consider. If you intend to upgrade Linux at some point, it may be convenient to have your development files on a separate partition. In this way, you can make a new file system for the new distribution, and you will not risk having large, unnecessary binaries which you no longer need. I like to have TEX and X386 on a separate partition, since these are not included in MCC distributions. You may like to put the `/usr` subtree on a partition of its own, or `/usr/src`. The directories `/home` and `/tmp` might also have partitions of their own, so that a user (including you) cannot accidentally fill up the system disk. See Section 7.2.2 [Dividing up your disk], page 37.

3.2. Swap space required for Linux

The amount of swap space you require depends on how much memory you have, and on what you intend to do on your system. What we call *swapping* in the Linux world is sometimes called *paging* on other operating systems. If you use the GNU C compiler, you may need 6 Mb of memory to compile one file. If you have only 4 Mb of physical memory, you will also need at least 2 Mb of swap space for that compilation. When GCC compiles a very large function with optimisation, it may need much more memory than this.

If you are using X windows, you may need at least 8 Mb of memory in addition to your other requirements. If you have several users (or if you are running large jobs in several windows or on several consoles), you must add up all their requirements for swap space.

On the other hand, most people are the sole users of their machines, and many of them do only one or two things at once. In such cases, you might plan on having a total of 8 to 12 Mb if you are not using X windows, or 16 to 20 Mb if you are using X windows and compiling big packages at the same time.

Linux can use two kinds of swap areas: *swap partitions* and *swap files*. Swap partitions can be used only for swapping. Swap files have the advantage of being easier to create and destroy, but they are less efficient than swap partitions. No swap partition or swap file can contain more than 16384 Kb of usable swap space, so if you need more than this, you must use several partitions or files.

MCC interim Linux can create and activate swap partitions during installation, and normally expects you to have a single swap partition, which it inserts into `/etc/fstab`, so that it is activated automatically whenever you boot the installed system. If you wish to have more than one swap partition, or a swap file in addition to or instead of a swap partition, you must create them yourself after installation and put the entries manually into `/etc/fstab`.

4. Installing or Updating MCC Interim Linux

This chapter assumes that you have already obtained all the necessary files and packages, and prepared or obtained a suitable MCC bootable root disk. See Section 2.1 [Getting MCC interim Linux by anonymous ftp], page 7, where the preparation of bootable root disks is described.

4.1. An overview of the MCC installation

In this section I summarise what you should do to install Linux using the MCC interim distribution. Exceptions, qualifications, and alternatives are omitted to keep the picture as simple as possible.

- Read this document carefully, so that you know what the process of installing Linux involves. Think about how you will partition your hard disk. See Chapter 3 [Preparing to Install Linux], page 11.
- Back up your hard disk.
- Get the boot image, packages, and DOS programs by using anonymous ftp in binary mode. See Section 2.1 [Getting MCC interim Linux by anonymous ftp], page 7.
- Unpack '`GZIP.EXE`'.
- Use the new '`GZIP.EXE`' to unpack the boot image.
- Use '`RAWRITE`' to copy the boot image to a high density floppy. See Section 2.1 [Getting MCC interim Linux by anonymous ftp], page 7.
- Boot from the MCC boot disk in read-only mode. See Section 4.2 [Booting from the MCC boot disk], page 14.
- Use '`fdisk`' to edit your partition table. See Section 7.2 [`fdisk`: the Linux partition table editor], page 36.
- Reboot from the MCC boot disk in read-only mode.
- Use '`mkswap`' to set up a swap partition. See Section 4.4.1 [Initialising a swap partition], page 18.
- Use '`mkfs`' to create file system partitions. See Section 4.4.3 [Creating a file system], page 19.
- Install the Base package from the MCC boot disk. See Section 4.5 [Installing basic files from the boot disk], page 20.
- Reboot from the MCC boot disk in read-write mode.
- Install the packages you select. See Section 4.6 [Installing the MCC interim packages], page 22.
- Use LILO to create a boot floppy. See Section 4.7.1 [Installing LILO on a floppy or hard disk], page 24.
- Reboot from your new boot floppy.
- Log in as root, and use '`/etc/lilo/install`' to install LILO on your hard disk.

The following sections explain this process in greater detail.

4.2. Booting from the MCC boot disk

Put the MCC boot disk in the 0 drive (the A: drive for DOS) of the machine on which you want to install Linux. Then reboot by pressing the 'RESET' button, or 'CTRL-ALT-DEL'. After the system boots, you should see something like this:

```
LILO
Press <RETURN> to boot with the floppy as a read-only '/'.
Type 'rw root=xxx <RETURN>' to mount a hard disk partition as '/'.
   The value 'xxx' is printed after you do an installation.
   Typical values:   /dev/hda1: 301     /dev/hda2: 302
                     /dev/hdb1: 341     /dev/hdb3: 343
                     /dev/sda1: 801     /dev/sdb4: 814
boot:
```

To install or update MCC interim Linux, you will need to boot at least twice from this floppy, and possibly three times:

1. You may need to run 'fdisk' to edit the partition tables. After doing this, it is a good idea to reboot the system.
2. You need to set up and mount the hard disk partitions and install basic files from the boot disk.
3. You need to reboot the system before installing the MCC packages.

In the first two cases, simply press <RETURN> after the above message appears. This boots Linux using the kernel on the boot disk, and mounts the boot disk as / in read-only mode.

In the third case, you need to give a command such as

```
rw root=302
```

where the number '302' varies depending on which root partition you intend to use. This also uses the kernel on the boot disk, but it mounts one of the partitions on your hard disk as /, and it mounts that partition in read-write mode. This is not guaranteed to work unless you have installed or updated basic files from the MCC interim boot disk.

As the kernel boots, it tries to explore your hardware, and messages will appear describing what it finds. If you have only one IDE disk, you will see this message:

```
hard disk I/O error
dev 0340, sector 0
  unable to read partition table of device 0340
```

This can be ignored; it means that the kernel cannot access your second hard disk, which of course does not exist. If you simply pressed <RETURN>, the messages which appear should end like this:

```
Linux version 0.99.pl8-5 (root@linux) 04/24/93 08:05:03

f   Run fdisk to manage the partition table.
s   Set up a new swap partition.
a   Activate an existing swap partition.
m   Make a new file system on a partition.
i   Install basic binaries.
c   Check (and repair) an existing file system.
e   Execute /bin/sh on an existing file system.
q   Quit.

? _
```

Instead of getting a shell prompt '#', you see a small menu. The choices in the menu help you execute the commands you need to use to create an installed MCC interim Linux system, and these commands appear on the menu more or less in the order in which you need to select them. To select a command, type the letter which appears next to it, and then press <RETURN>. This procedure differs from that in previous MCC installations.

4.3. Editing the partition tables

To run 'fdisk' from the MCC boot floppy, select 'f' from the menu. See Section 7.2 [fdisk: the Linux partition table editor], page 36, for more information about the 'fdisk' command. First you need to specify the disk whose partition tables you will edit. The menu utility asks:

```
Run fdisk on (block device; default = /dev/hda):
```

At this point, press <RETURN> to accept the default. /dev/hda is the first hard disk, /dev/hdb the second, /dev/sda the first SCSI disk, /dev/sdb the second, /dev/xda the first XT disk, and /dev/xdb the second. To select one of these, simply type it after the prompt. If you decide not to, the 'block device' prompts always accept 'q' as an answer, though sometimes the message 'That is not a legitimate device' appears.

At this point the 'fdisk' prompt appears:

```
Command (m for help): _
```

If you type 'm', you will see the main menu for 'fdisk':

```
Command action
   a   toggle a bootable flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   p   print the partition table
   q   quit without saving changes
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help): _
```

We can give the command 'p' to print the partition table. Here is a sample from one of the machines I use:

```
Disk /dev/hda: 14 heads, 35 sectors, 978 cylinders
Units = cylinders of 490 * 512 bytes

   Device Boot  Begin   Start     End  Blocks   Id  System
/dev/hda1   *       1       1      98   23992+   4  DOS 16-bit <32M
/dev/hda3   *     263     263     426   40180   81  Linux/MINIX
/dev/hda4         427     427     977  134995    5  Extended
/dev/hda5         427     427     491   15924+  82  Linux swap
```

We can create a primary partition in slot 2, or another logical partition after the swap space:

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (99-978): 99
Last cylinder or +size or +sizeM or +sizeK (99-262): 262

Command (m for help): _
```

Then we change its type from 81 (Linux/Minix) to 83 (Linux extended file system). This actually makes no practical difference, but I like to see things clearly when I look at partition tables; I want sensible information to appear.

```
Command (m for help): t
Partition number (1-8): 2
Hex code (type L to list codes): 83
Changed system type of partition 2 to 83 (Linux Ext)
```

Next we make the new partition active. In most cases this has no practical effect, but I like to be able to see which partitions are bootable when the partition table is printed:

```
Command (m for help): a
Partition number (1-8): 2

Command (m for help): p

Disk /dev/hda: 14 heads, 35 sectors, 978 cylinders
Units = cylinders of 490 * 512 bytes

   Device Boot  Begin   Start     End  Blocks   Id  System
/dev/hda1   *       1       1      98   23992+   4  DOS 16-bit <32M
/dev/hda2   *      99      99     262   40180   83  Linux Ext
/dev/hda3   *     263     263     426   40180   81  Linux/MINIX
/dev/hda4         427     427     977  134995    5  Extended
/dev/hda5         427     427     491   15924+  82  Linux swap
```

Being satisfied with this, we decide to save it.

```
Command (m for help): w
The partition table has been altered!

Calling BLKRRPART ioctl() to re-read partition table
Syncing disks
Reboot your system to ensure partition table is updated
Reboot the system now (Y or N): y

Leave the boot floppy in the drive.

Press <RETURN> at the LILO prompt.
```

It is not strictly necessary to reboot the system at this point. It is perfectly safe to run '`fdisk`' again, whether on the same partition or on other partitions. But before you run any other commands from the boot floppy, it is safer to reboot, at least if the hard disk contains any data which you would regret losing.

4.4. Preparing swap and file system partitions

Once partitions have been created, they need to be initialised before they can be used for swapping or as file system partitions. Previously initialised swap partitions should be activated during installation, and it is a good idea to check existing file systems so that any inconsistencies that may have arisen are corrected before you copy new files onto them. You can select any one of these functions from the menu which appears after you boot the system in read-only mode.

4.4.1. Initialising a swap partition

Before you can use a swap partition, you must initialise it. Once Linux is installed, you can do this with the '`mkswap`' command. To run '`mkswap`' from the MCC boot floppy, select '`s`' from the menu. Here is a typical example:

```
WARNING: This will destroy any files on this partition.
Do you want to continue (Y or N): y
Run mkswap on (block device; default = /dev/hda5):
Setting up swapspace, size = 16302080 bytes
Adding Swap: 15920k swap-space
```

I find warnings of the type 'Are you sure?' irritating, but I print one here to remind you that this command overwrites an entire partition. The default swap device is `/dev/hda5` because this is the first logical partition, and it seems sensible not to put a swap partition in one of the slots of the primary partition table, unless you have a very small disk. If you are using a different partition for swap, type its name before pressing `<RETURN>`; for example, type `/dev/hda2` in response to the 'block device' query.

The maximum usable size of a swap partition is 16 Mb, or 16384 blocks of 1024 bytes each. If you try to initialise a partition larger than this for swap, you are warned that some of this space will be wasted, and you can choose to give up before doing so. If you wish, you can use '`fdisk`' to reduce the size of the partition slightly, and you can add the extra cylinders to another partition.

After the new swap partition is initialised successfully, the boot utility activates it for you.

4.4.2. Activating a swap partition

If you install Linux on a disk which already contains an initialised swap partition, you should activate it before doing anything else. On small systems, the installation may fail if you do not, and on any system, this should make the installation a little faster. Once Linux is installed, you can activate swap space with the '`swapon`' command. To run '`swapon`' from the MCC boot floppy, select '`a`' from the menu. Here is a typical example:

```
Activate swapping on parition (block device; default = /dev/hda5):
Setting up swapspace, size = 16302080 bytes
Adding Swap: 15920k swap-space
```

Here again, the default swap partition is `/dev/hda5`.

4.4.3. Creating a file system

Creating a file system under Linux is like running the `FORMAT` utility under DOS. Once Linux is installed, you can do this with the '`mkfs`' command. To run '`mkfs`' from the MCC boot floppy, select '`m`' from the menu. Here is a typical example:

```
WARNING: This will destroy any files on this partition.
Do you want to continue (Y or N): y
Run mkfs on (block device; default = /dev/hda1): /dev/hda2
Create a Minix filesystem (Y or N): n
By default all 40180 blocks of this partition are used for the file system.
Sometimes this causes problems, due to a bug in mkfs.ext2.
Reduce this number (Y or N): n
mke2fs 0.3, 93/04/22 for EXT2 FS 0.3, 93/04/22
```

MCC interim Linux currently supports two kinds of file system: the *minix* file system and the *ext2* file system.

The *minix* file system is named after the MINIX operating system. It comes in two slightly different versions: one in which file names cannot be more than 14 characters long, and one in which file names cannot be more than 30 characters long. Partitions with the 14 character limit are actually compatible with MINIX and can be mounted under that system. The 30 character limit is not compatible with MINIX, but Linux treats the two versions as identical in most respects. The *minix* file system is older and less subject to change, and perhaps slightly more reliable. We do not recommend using the 14 character version unless you need to maintain compatibility with the MINIX operating system. A *minix* file system cannot have more than 64 Mb, or 65536 blocks of 1024 bytes, on a single partition.

The *ext2* file system was developed for Linux and may eventually become the primary Linux file system. An older version of this system, the *ext* file system, is now quite stable, but we do not recommend its use, and MCC interim Linux does not support it any longer. File names in the *ext2* file system may be up to 255 characters long, and a single partition may contain up to 4 terabytes (4 million megabytes). If you tell the MCC installation utility that a file system is not a *minix* file system, it assumes that it is an *ext2* file system.

4.4.4. Checking and repairing an existing file system

It is a good idea to check file systems once in a while, particularly before you install a new version of Linux on a file system which already exists. Once Linux is installed, you can do this with the '`fsck`' command. This is the Unix equivalent of the DOS command '`CHKDSK`'. To run '`fsck`' from the MCC boot floppy, select '`c`' from the menu. Here is a typical example:

```
Run fsck on (block device; default = /dev/hda1): /dev/hda2
Does this device contain a Minix file system (Y or N): y
Do you want (possibly dangerous) automatic fixing (Y or N): n
Do you want to fix problems on request (Y or N): y
```

In the same way as when you create a file system, the boot floppy assumes that a Linux file system is either a *minix* file system or an *ext2* file system. The '`fsck`' program reads through a file system, which is usually described in some detail and with a considerable amount of redundant information. If the information is inconsistent, '`fsck`' does one of three things:

1. In *automatic* mode, it tries to fix all problems as well as it can.
2. In *interactive* mode, it asks you before doing anything.
3. In *default* mode, it fixes minor problems, but asks you before fixing major problems.

If you answer '`y`' to the query about automatic fixing, '`fsck`' runs in automatic mode and fixes everything it can. This may be dangerous if the file system is badly corrupted, since the program might delete a file you wish to keep. In my view, this is very unlikely, and I nearly always run '`fsck`' in automatic mode. If you answer '`y`' to the query about fixing problems on request, '`fsck`' runs in interactive mode; otherwise, it runs in default mode.

File systems usually become corrupt when a system is shut down or rebooted improperly; for example, while a file or directory is being created or deleted. Chances are that if a file is being written, you will lose all or part of the file irrevocably.

4.5. Installing basic files from the boot disk

When you have set up the swap and file system partitions, you can select the *install* option from the menu with the `i` command. The following dialogue shows how to specify partitions to be mounted:

```
Which device contains the file system you wish to be mounted
on '/'?  Give a name in the form '/dev/hda1', or type
'q' to quit.
This file system is on (block device; default = /dev/hda1): /dev/hda2
Does this device contain a Minix file system (Y or N): n

Is there another partition you wish to mount?  If so, give the
pathname it will have in the installed system, such as '/usr/src'
with a slash at the front and no slash at the end.  (Upper case is
not allowed.)  If not, type 'n': n
```

You must always give directories moving away from the root of the tree: in other words, if you intend to mount `/dev/hda8` on `/`, `/dev/hda7` on `/usr`, and `/dev/hda6` on `/usr/src`, then you must give `/` first, then `/usr`, and then `/usr/src`. If you specify another directory here, for example, `/usr/src`, then you are asked for the block device where it lives, and the type of file system it contains.

After you type 'n' to the request for another directory, you should see some messages as the *Base* package gets installed:

```
499 blocks
Installing Base.tgz
Base contains basic binaries and text files for Linux.
...................................................................
1971 blocks
96 blocks
Verifying Base.crc.
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/hda2               40180    2272     37908      6%   /root

Reboot the system now (Y or N): _
```

The 'blocks' messages refer to 512-byte blocks, half the size of the file system blocks. 'Verifying' refers to the crc check by 'brik'; after checking, the crc file is moved to /tmp, so that you can check later to see whether anything has changed. There is no verification for most man pages, or for files (like /etc/fstab, for example) which are edited by the installation programs. If a file has a bad checksum, a message appears like this one:

```
BAD etc/init
```

This warns you that something is wrong. If there is a small error (caused perhaps by a bad spot on your floppy?), you might have only one bad file, but usually there are several. The 'df' listing (which includes all mounted disks) can be useful if you have insufficient space to install further packages.

It is in general a good idea to reboot the system at this point during installation or upgrading. If you choose not to do so, a shell is started, and you can poke about in the half-installed system. If you choose to do so, you will see this message:

```
Please type CTRL-D to reboot, not 'reboot' or 'shutdown'.
```

This is because the system is set up in an odd way. The root partition (the booted floppy) is mounted in read-only mode, and the usual 'mount' and 'umount' programs will not work properly. Similarly, the 'reboot', 'halt', and 'shutdown' utilities are not able to unmount file systems, which consequently are likely to become corrupt during the shutdown. If you type CTRL-D or 'exit', you will again see the prompt

```
Reboot the system now (Y or N): _
```

If you answer 'y', you should see a message like this:

```
Leave the boot floppy in the drive.
Unmounted /dev/hda3
Unmounted /dev/hda2
```

```
Type 'rw root=302' at the LILO prompt.
```

The message '`rw root=302`' is important, since you need to type it after the system has rebooted and printed the LILO prompt. The message printed at that time will jog your memory if you forget this, or if you did not write it down during the 15 seconds or so while it appears on the screen. The important part of the message is the number '`302`', which varies depending on the partition you wish to be mounted as `/`, the root of the new file system.

4.6. Installing the MCC interim packages

After you have installed the *Base* package from the boot floppy, you must reboot before you install any other packages. At this time you should not yet be able to boot Linux from your hard disk. Leave the boot floppy in the floppy drive, and when the LILO prompt appears, type the command '`rw root=xxx`' which was printed at the end of the last step. The number '`xxx`' is a hexadecimal code for the partition which contains the root of the file system, and which is mounted as `/`.

The system should boot, mounting the specified partition as `/` in read-write mode. Then the following message should appear:

```
Which drive will you use to install Linux packages:

1  drive A:  5.25 inch HD floppies
2  drive A:  3.5  inch HD floppies
3  drive B:  5.25 inch HD floppies
4  drive B:  3.5  inch HD floppies
5  A directory on an unmounted partition on this machine
6  A directory already mounted at this time
7  No installation at this time

? _
```

If you are installing from floppies, give one of the answers 1 – 4, whichever is appropriate. If you are installing from a directory, and if you mounted that directory before you installed the Base package from the boot floppy, answer '`6`'. If you copied the packages to a directory on a DOS partition, or on a Linux partition which is not currently mounted, answer '`5`'.

If you answer '`5`', you must give the name of the unmounted partition which contains the directory which contains the packages. The partition must be a DOS partition or contain a *minix* or *ext2* file system. You are prompted for the name of the partition:

```
What is the block device on which this partition lives?
```

To this you should answer, for example, `/dev/hda1`.

If you are going to mount a partition using NFS, then you must initialise your network before you continue. If you must install the *tcpip* package to do this, then give one of the answers 1 – 6 as appropriate, and install the *tcpip* and *base* packages first. But if your network software is already installed, so that you got messages when the system booted about your hostname and IP address, type '7'. In this case, it may be necessary to change your hostname and run `/etc/rc.net` before you continue. Once the hostname is correct and networking is active, mount the remote partition on `/mnt` and give the command '`/tmp/install`'. This is done automatically if you give one of the commands 1 – 6 at boot time.

Whichever method you use to make packages accessible, you must then answer this question:

```
Do you wish to have a prompt before each package (Y or N): _
```

I recommend answering '`y`', which allows you to omit packages which you do not wish to install. On the other hand, when I install Linux on several machines at once, I make certain the disks contain only packages which I wish to install on all machines, and I answer '`n`' at this point. If you are confident that only desirable packages will be installed, and that you have enough disk space to hold them, answer '`y`', then drink some tea while the packages are installed. At this juncture you may see all, some, or none of the following dialogue, depending on the options you have chosen before this point:

```
If you wish, you may install other parts of MCC interim Linux
from a directory on another machine, remotely mounted using NFS,
or from a currently available directory mounted in some other way.
This will only work if you have mounted the remote directory
If the directory is not accessible now, do you wish to quit? (Y or N): n
Do you wish to install from a currently mounted directory? (Y or N): y
/dev/hda1 is mounted on /mnt.  In which directory are the packages found?
/mnt/mcc
```

After you reach this point, packages from the selected directory or floppy are installed. The messages which appear are much like those for the *Base* package on the boot floppy, and include the following information:

```
Base contains basic text and binary files, with formatted man pages.
Bison contains the GNU yacc-compatible parser generator.
Emacs contains GNU emacs 18.59, except for info files.
Flex contains the GNU clone of lex, the lexical analyser generator.
Gawk contains the GNU version of the awk programming language.
Gcc contains the GNU C compiler, release 2.3.3, excluding g++ and gprof.
Gdb contains the GNU debugger.
Gpp contains the GNU C++ compiler, release 2.3.3, except for those
    files contained in the gcc package.
Gprof contains the GNU profiler gprof and its associated library files.
Groff contains the GNU clone of the nroff/troff text formatter.
Info contains GNU info, makeinfo, and selected info files.
```

```
Kermit contains the Kermit file transfer and serial communications program.
Linux contains the kernel source files for Linux version 0.99-p8+.
Patches contains all patches used to compile the binaries included in the
    current MCC interim distribution.
Tar contains the GNU version of tar, the tape archive program.
Tcpip contains networking programs, daemons, and configuration files.
    These will not work unless your kernel was compiled with tcpip.
The kernel in ide-uk has no TCP/IP, NFS, SCSI or xd support.
The kernel in ide-us has no TCP/IP, NFS, SCSI or xd support.
The kernel in ipide-uk has no SCSI or xd support.
The kernel in ipide-us has no SCSI or xd support.
```

If you have asked to be prompted, the message is printed before the prompt, so that you know what you are accepting or rejecting. During a floppy installation you will also see the following message from time to time:

```
You may now remove the floppy disk from the drive.  If you wish
to install from another disk, then put it in the drive and
type 'r' when ready; or type 'q' to stop: _
```

If your disks are DOS disks, you should have no other messages at mount time except '`VFS: Disk change detected on device 2/28`', or some such message, depending on your disk drive. But if your disks contain *minix* or *ext2* file systems, you will get some error messages. I allowed these to remain, since they might sometimes contain useful information.

The installation concludes by giving you the opportunity to install LILO either on a floppy or on your hard disk. If you do not wish to install LILO, you may create a conventional boot floppy instead. If you do not create a boot floppy or install LILO on your hard disk, you will not be able to boot your newly installed system, unless, of course, you use some boot loader which is not included in the MCC interim distribution.

4.7. Creating a boot floppy

After you install the last of the MCC packages, typing '`q`' if necessary to say that you have no more floppies, the installation procedure allows you to create a boot floppy, using either LILO or your newly installed kernel. You may also install LILO on your hard disk if you wish, though it is safer to make a boot floppy first.

4.7.1. Installing LILO on a floppy or hard disk

LILO is Werner Almesberger's utility for booting Linux: it stands for *LInux LOader*. It is quite versatile. You can install it in the Master Boot Record on your hard disk, so that it functions as the primary boot program when your system boots, or you can install it in any partition on your hard disk, or on a floppy disk. LILO can allow you to select from up to 16 possible Linux boot images, or you can boot another operating system. It can pass special options to Linux kernels, such as an instruction to boot in single user mode, or to mount the root partition in read-only mode. It can boot a default option unless you intervene during boot, and it can force you to choose one of its images interactively. It can also protect some or all of your boot options by requiring a password at boot time.

The MCC *base* package (not the Base package on the boot disk) installs LILO in the directory `/etc/lilo`. Among the files installed in this directory are a detailed `README` which describes some of the many features of this utility, the command '`/etc/lilo/lilo`', which installs the loader, and the command '`newinstall`', which is a simple utility for installing LILO. The '`newinstall`' utility is run from the MCC installation procedure, which asks you the following questions:

```
LILO can be used to boot Linux from your hard disk.  It
can also give you the choice of booting one or more versions
of Linux or some other operating system.  LILO can install
itself in any partition, or in the Master boot record, or on
a floppy disk, if you wish to be safe.  Would you like to
install LILO now? (Y or N): y

Where should the boot code go?  You may override this later to write a
test floppy before installing LILO on the hard disk.  Possible choices
include /dev/hda (the master boot record on your first hard disk),
/dev/hda3 (the partition you are installing Linux on),
or /dev/fd0 (floppy disk 0, DOS floppy A:).
Give me the name of a block device.  The default is /dev/hda:
```

The default at this point is chosen by taking the name of your root partition (for example, '`/dev/hda3`'), removing the end of its name (leaving '`/dev/hd`'), and adding '`a`' (thus producing '`/dev/hda`' in this case). I recommend accepting this default.

```
LILO will boot by default the first system you specify.
Others must be selected by taking special action at boot time.
Is the first system you wish to add a version of Linux or not?
Answer 'n' for other operating systems or 'q' to quit:  (Y, N, or Q): y
The default Linux image in this installation is /etc/Image.
Press <RETURN> if you are using this image, or if you have a
different image somewhere else which is currently mounted,
give its pathname:

What would you like to call this option?  The default is 'linux'.
Remember, you may not give the same name to two options:

Is the next system you wish to add a version of Linux or not?
Answer 'n' for other operating systems or 'q' to install:  (Y, N, or Q): n

In what partition does this other system live?  Press <RETURN> if
you do not wish to add this option.  Otherwise give an answer
like /dev/hda1: /dev/hda1

What is the name of the device where this partition lives?
I want the block device which is the whole of the disk.
Press <RETURN> to accept the default /dev/hda, or 'q' to give up:
```

```
What would you like to call this option?  The default is 'dos'.
Remember, you may not give the same name to two options:

Is the next system you wish to add a version of Linux or not?
Answer 'n' for other operating systems or 'q' to install:  (Y, N, or Q): q
```

This little loop repeats, expecting the answer '`y`' for a Linux image, '`n`' for another operating system, and '`q`' when you have no more options to add. You must be careful not to give the same name to two options. The file `/etc/Image` is installed by the MCC boot floppy. When your list is finished, and if you have not added an option named '`linux.old`', '`newinstall`' inserts such an option to boot an old kernel named `/etc/Image.old` if it exists when LILO is installed. The MCC optional kernel packages produce such a file, and the script '`/usr/src/linux-mcc/install`' produces such a file whenever you compile a new kernel.

```
Now where would you like to install LILO at this time?  You may put
it on /dev/fd0 to be safe; be sure you have a formatted floppy disk
in this drive NOW, and that it is a device you can boot from, not
your DOS B: drive.  Or you may install it in /dev/hda.
Give me the name of a block device.  The default is /dev/fd0: /dev/fd0
Added linux
Added dos
Added linux.old

This release of MCC interim Linux has now been installed.
You may now login as root or as user.
```

This installation of LILO may fail if you have an odd disk, whose geometry LILO is unable to determine when it is run. In that case, you can log in, edit the file `/etc/lilo/disktab` to insert the correct geometry for your disk, then give the command '`/etc/lilo/newinstall`' to try again.

The boot floppy produced at this point will allow you to test whether LILO works properly before you attempt to install it on your hard disk. If you do install LILO on your hard disk, and for some reason it fails, or you simply wish to remove it, your original Master Boot Record is saved in the file `/etc/lilo/boot.xxxx`, where '`xxxx`' is a hexadecimal number based on the drive number. You can restore this with the command '`dd if=/etc/lilo/boot.0300 of=/dev/hda`', using the appropriate names for your system. You should be very careful: the saved Master Boot Record restores everything to its original state, including the first four partitions described in the primary boot sector.

4.7.2. Making a conventional boot floppy

If you do not install LILO after installing the MCC packages, you are asked whether you wish to make a conventional boot floppy.

```
You can now make a conventional boot floppy which will boot
your newly installed Linux system.  To do this, you must put a
formatted high density diskette into your A drive and answer 'y'.
```

```
If you wish to do this now, answer 'y'. (Y or N): _
```

If you answer 'y', this simply copies your kernel to the floppy in drive 0 using the command

```
/bin/dd if=/etc/Image of=/dev/fd0 bs=3072
```

The resulting disk should boot under any circumstances. Thus, if you cannot get LILO to work, or if you prefer not to use it, you have a way of rebooting your system without messing about with the partition tables. This should not really be necessary. There are very few known problems with LILO other than the need to put correct parameters in `/etc/lilo/disktab` for a few awkward disks.

5. Recovering from Fatal Disasters

You can use the MCC boot floppy to recover from fatal disasters. What kind of disaster requires such measures? Sometimes people accidentally do things which make the system unbootable. For example, they may edit or delete one of the configuration files in `/etc`, or they may delete a fundamental utility, like `/etc/init` or one of the shared libraries in `/lib`.

In many of these cases, the system will still boot in single user mode. To do this you must have installed LILO as your boot loader, even if it is installed in a file system partition and is called by another boot loader, such as the DOS or OS/2 loaders. When LILO boots, you must force it to prompt you. No special action is necessary if you put the '`prompt`' command instead of the '`delay`' command in `/etc/lilo/config`; this forces a prompt in all cases.

To force LILO to prompt, you must perform one of the following actions:

- Hold the '`SHIFT`' key down at boot time.
- Hold the '`CTRL`' key down at boot time.
- Hold the '`ALT`' key down at boot time.
- Set the '`CAPS LOCK`' key on at boot time.
- Set the '`SCROLL LOCK`' key on at boot time.

When the prompt '`boot:`' appears, you must type the name of your default kernel followed by the word '`single`'; for example,

```
boot: linux single
```

If you don't remember the name of your kernel, press the '`TAB`' key. This causes a list of the available options to be displayed.

In very serious cases, when you cannot boot in single user mode, you may use the MCC boot floppy to recover. Simply boot from the floppy in read-only mode, pressing '`<RETURN>`' at the LILO prompt. If the file `/bin/sh` is intact on your hard disk, you can choose '`e`' from the menu, and this will start a shell after mounting the various partitions. At this point, most of the editors and other utilities should work correctly. The commands '`mount`' and '`umount`' do not work at all. Repair the damage, then reboot the system. Do not use '`halt`', '`reboot`', or '`shutdown`' to reboot; type '`CTRL-D`' or '`exit`'.

In the very worst possible case, for example when `/bin/sh` has been deleted, you can re-install Linux from the boot floppy. As with all MCC upgrades, this creates a special directory named `/backupdirs`, in which your existing configuration files are saved; for example, `/etc/passwd` is moved to `/backupdirs/etc/passwd`. It is not possible — or desirable — to leave such files if they exist, since the newly installed system might not boot. The file `/etc/syslogd` is also moved to the `/backupdirs` directory when you install the *Base* package from the boot floppy. You will need to run LILO again before rebooting. Use the command

```
ROOT=/root;/etc/lilo/lilo
```

If you recompile your kernel, or if you use one of the smaller MCC kernels, you may wish to customise the MCC boot floppy. Assuming that you have your kernel in `/etc/Image` and the matching data file `/etc/psdatabase`, mount a 'standard' MCC boot floppy in write mode from drive 0 on `/mnt`. Then give these commands:

```
cp -p /etc/Image /mnt/etc/Image
tar cf - /etc/psdatabase|gzip -9 >/mnt/psdata.tgz
/etc/lilo/lilo -C /mnt/etc/config.lilo
umount /mnt
```

This should produce a recovery system that will boot your own kernel (so that the '`ps`' commands all work) and reinstall your own kernel if you install Linux from the floppy.

6. Tailoring MCC Interim Linux to Your Taste

There are several changes which you might make to your newly installed Linux system, whether to make it run better on your machine, or to adapt it to suit your personal taste. Some of these changes are discussed in the following sections of this document.

6.1. Recompiling the kernel

There are many good reasons why you might wish, or even need, to recompile the kernel. You may wish to remove or include drivers or file systems, to add some of the available patches, to support your own national keyboard, to remove unnecessary features such as maths coprocessor emulation, or to add support for more than 16 Mb of RAM.

To recompile the kernel, you should install the *gcc* and *linux* packages from the MCC distribution. Then 'cd' to /usr/src/linux-mcc. Edit the Makefile to choose your national keyboard, default SVGA mode, etc, as described in the README file in that directory. Then type

```
make config
make dep
make
./install
```

The last command installs the newly compiled kernel in /etc and uses LILO to prepare to boot from it. Reboot, login as root, and type the command 'psupdate' immediately. This changes the file /etc/psdatabase so that the 'ps' commands work with the new kernel. The 'install' script also saves the old /etc/Image as /etc/Image.old. This should allow you to boot using LILO with the option 'linux.old' if you have any difficulty booting with the new kernel.

Note: Several people have commented on the fact that this release puts the include files in /usr/include/asm and /usr/include/linux in the directory /usr/include rather than in /usr/src/linux(-mcc). This is actually more convenient if you are not recompiling the kernel repeatedly. It means that (1) the C compiler works without installing the kernel sources, and (2) it continues to work after you delete the kernel sources. Nevertheless, if you are doing a lot of work with different kernels, you may find it more convenient to move these two directories and their contents to /usr/src/linux/include, and put symbolic links to them in the /usr/include directory:

```
mkdir -p -m 755 /usr/src/linux /usr/src/linux/include
(cd /usr/include;tar cf - asm include)|(cd /usr/src/linux/include; \
 tar xvfp -)
rm -rf /usr/include/asm /usr/include/linux
ln -s /usr/src/linux/include/* /usr/include
```

6.2. Setting up Western Digital Ethernet cards

The 'standard' kernel as distributed by Linus Torvalds currently contains support only for Western Digital Ethernet cards. These should be installed with the two jumpers set to '`soft`'. It is a good idea to have the DOS program '`EZSETUP`' to hand, so that you can configure the card that way in case '`wdsetup`' doesn't work. The file `/etc/inet/README.wdsetup` is installed as part of the *tcpip* package, and contains more information about this program.

Login as root, then '`cd /etc/inet`'. You should configure the card before starting the networking software; otherwise you must (after issuing '`shutdown -h`', of course) press the '`RESET`' button or switch the power off and on before the changes take effect. Give the command

```
./wdsetup -a 280
```

to set the card's address to an appropriate value. If this doesn't work, you need to read the file `README.wdsetup` for more information.

6.3. Configuring the *tcpip* package

The TCP/IP programs included in this release of MCC interim Linux will not work unless you use a version of the kernel which includes TCP/IP and, if you wish, NFS. The default kernel on the boot disk does include both TCP/IP and NFS. This software should work even if you have no Ethernet card, though you will need to comment out the last line beginning with '`$CONFIG`' in `/etc/rc.net`. In that case, of course, you cannot contact other machines.

The kernel source supplied, like the compiled kernels, contains only the Ethernet drivers contained in Linus Torvald's official release of the kernel. If you want support for one of the other Ethernet cards, you must get patches from somewhere else and install them before recompiling the kernel. A recent version of these patches is available by anonymous ftp from `ftp.mcc.ac.uk` in the directory `/pub/linux/newether`.

After installing the *tcpip* package, give the command '`/etc/install.net`'. The following dialogue ensues:

```
# /etc/install.net
Enter IP Address for avl0 (aaa.bbb.ccc.ddd):130.88.201.62
Enter Net Address for avl0 (aaa.bbb.ccc.0):130.88.201.0
Enter Router Address for avl0 (aaa.bbb.ccc.ddd):130.88.200.251
Enter Domain Name for avl0:mcc.ac.uk
130.88.201.62 avl0.mcc.ac.uk avl0
Name Server for Domain mcc.ac.uk (aaa.bbb.ccc.ddd):130.88.200.6
You may add more nameservers by editing /etc/resolv.conf
```

Here instead of '`avl0`' your hostname should appear, and you must give your own addresses instead of '`130.88.x.x`', and your own domain instead of '`mcc.ac.uk`'. Before you try to run any

TCP/IP software, you might need to run '`wdsetup`'. See Section 6.2 [Setting up Western Digital Ethernet cards], page 32.

After installing *tcpip*, you should edit the file `/etc/inetd.conf` to uncomment any services you wish to allow. To enable a service, remove the initial '`#`'. If '`inetd`' is already running, you must send it SIGHUP or reboot after changing this file; otherwise the changes have no effect. Some of these services, particularly `tftp`, may expose your machine to crackers if you have a network connection. Read the man pages and be cautious!

If you have no local domain name server, you may wish to delete the file `/etc/resolv.conf` and to remove '`bind`' from the file `/etc/host.conf`.

I have supplied the '`named`' daemon, but I make no attempt to teach beginners how to set it up! There is some information in the man page for `named`, but it is very sparse and needs to be filled out with more comprehensive examples. In any case, it should not be necessary to run '`named`' in most installations.

Note: These programs are compiled from the most recent source I could find. I have fixed several bugs, but they are still *buggy*. I have included them in the release partially because so many people ask for them, and partially because I use them myself.

6.4. Editing the system configuration files

After installing this version of MCC interim Linux, you may wish to customise it. The following files should probably be edited:

`/etc/rc.local`
: This contains the command which sets your host name. Change '`linux`' to the actual hostname of your machine. This is important if you intend to use TCP/IP.

`/etc/mtools`
: See Section 7.4 [Special commands for DOS floppies and partitions], page 46.

`/etc/passwd`
: If you wish to add a new user, edit this file. The format is

```
pw_name:pw_passwd:pw_uid:pw_gid:pw_gecos:pw_dir:pw_shell
```

where the fields have the meanings defined in the file `/usr/include/pwd.h`: user name, encrypted password, user ID number, group ID number, real name, login directory, login shell.

`/etc/lilo/config`
: This contains the command which controls what happens whenever LILO is reinstalled. You must give the command '`/etc/lilo/install`' whenever the system file (`/etc/Image` by default) is modified (for example, by the '`rdev`' command) or moved: LILO does not use the directory structure to find the file.

`/etc/lilo/disktab`
: If you have one of the SCSI disks for which Linux cannot get the geometry, you must edit this file before installing LILO; otherwise the installation will fail.

`/etc/inetd.conf`

All services for inetd are commented out. See Section 6.3 [Configuring the *tcpip* package], page 32.

No other files should *need* to be changed.

7. Using Linux

7.1. Caring for your Linux system

Linux is an operating system which must be handled more carefully than DOS, or at least older versions of DOS. There are some important facts about Linux which will help you to have fewer problems and get more out of your system.

Linux has several file systems, but all of the 'native' Linux file systems have one feature in common: they require special care which a DOS file system does not. First, information is not written to disk as soon as it becomes available, but it is flushed at regular intervals (every 30 seconds by default). This improves performance. But you must **never** switch off a PC which is running Linux, since some data may not get written to disk, and will be lost, and the data describing the file system may become inconsistent, putting the files in the system at risk of corruption.

To shut Linux down safely, there are several methods you might use:

1. Give the command '`sync`' before switching off. This forces the system to flush unwritten buffers. To be a little safer, give the command twice, and wait for it to finish.
2. Reboot the system either by pressing '`CTRL-ALT-DEL`', or by giving one of the commands '`reboot`' or '`shutdown -r`'. This is better than the '`sync`' method.
3. Stop the system by giving one of the commands '`halt`' or '`shutdown -h`'. This is the safest method, but it can only be done by the user 'root'.

For the same reason, floppy disks may be corrupted if you remove them from the drive before unmounting them. Should this happen, you can repair them with the command '`fsck`'. MCC interim distributions are configured to run '`fsck`' on all mounted file systems at boot time, unless you are rebooting after having given the command '`shutdown`' with the '`-f`' option.

There are a few files, system logs and such, which grow without limit. The logs in `/usr/adm`, particularly the file `messages`, are restarted whenever the system is booted, so they should not cause problems unless you leave the system running for long periods, or when the file system is nearly full. But the file `/etc/wtmp` will grow constantly, and you may need to shorten it. To do this, log in as root and give the command

```
> /etc/wtmp
```

If you install networking software and connect your machine to the outside world, you may be exposing yourself to unscrupulous hackers. Be careful to set passwords, particularly for the root user, using the command '`passwd`'.

The superuser '`root`' has very great powers, and can easily damage or delete the entire system. It is a good idea to do most of your work as 'user' (or as some other user name), and to become root only when you need to run one of the commands which only root can give. For example, you can make a file read-only by using the '`chmod`' command, so that a simple '`rm`' command will not delete it, but '`rm`' will not normally ask the root user for confirmation before deleting such a file.

To learn more about Unix systems, get a good book, read it, and follow some of the advice it gives.

7.2. `fdisk`: the Linux partition table editor

`fdisk` is the Linux partition table editor. In this section we examine this utility and try to describe it thoroughly enough so that anyone can use it.

7.2.1. Disks and how they are described

A typical disk consists physically of one or more circular objects called *platters*, which rotate about a central axis. Devices called *heads* move to specified places on the disk surface to read or write information. There is usually one head on each side of every platter, and all these heads are attached to a comb-like controller arm which moves all of them at the same time, either closer to the centre of the disk, or closer to the outer edge.

Suppose the arm is in one position, putting an area of the disk surface within reach of one or another of the heads. This total area, everything that is accessible without moving the arm, is called a *cylinder*. (A cylinder is a barrel-shaped cross section of a disk, consisting of a circular strip from each side of each platter.) The part of a cylinder that one head can read or write without moving is called a *track*.

Each track is divided into several pie-shaped slices called *sectors*, which are the smallest parts of the disk which can be read or written at a time. The sectors on one disk are usually all the same size.

In fact, there are not always two heads to every platter, there are some disks which do not have the same amount of data in every cylinder, and there may be disks which do not have the same amount of data in every sector. These features are usually hidden on PCs by the controller card or the BIOS, which map the physical geometry of a disk onto a logical geometry, which is what is actually used to access the disk.

The numbers which describe the *geometry* of a disk are

1. The number of cylinders it contains.
2. The number of tracks per cylinder, which is the number of heads.
3. The number of sectors per track.
4. The number of bytes per sector.

These numbers vary from disk to disk, but a typical PC disk might have about 1000 cylinders, half a dozen heads, and 15 or 20 sectors per track, with each sector containing 512 bytes or characters; such a disk contains 40 to 60 megabytes of data. A *double density* floppy disk contains 40 cylinders, with 2 heads (2 tracks per cylinder), and with 9 sectors per track; such a disk contains 360 kilobytes, or 360 * 1024 characters. A *high density* 3.5 inch floppy contains 80 cylinders, with 2 heads and 18 sectors per track, or 1.44 megabytes, or 1440 * 1024 characters.

The exact size of a track or cylinder in bytes varies from one disk to another. This `fdisk` for Linux deals mainly with cylinders, since this is the best unit to use when allocating space for partitions. It reports partition sizes in *blocks* of 1024 bytes, or 2 sectors, since '`mkswap`' and the various '`mkfs`' programs require this number. A block is the smallest amount of space which can be set aside for a file in the current file systems.

An operating system, such as Linux or DOS or OS/2, may use a disk in any way that it wishes, but if two operating systems share the same disk, they must agree on who owns what, or else one will interfere with the other (that is, by damaging the other's files). A *partition* is a section of a hard disk which is handled as a unit by all operating systems which can access the disk. The standard way to define partitions (for the moment) is the *partition table*, a list of information which is stored in parts of the disk that don't belong to any of the systems using the disk. The beginning of the partition table is stored in the disk's primary boot sector, and the rest is stored in a chain of sectors scattered throughout the disk.

The first sector on the disk is called the *primary boot block* or *primary boot sector* because (1) it comes first, before other, similar sectors; (2) it tells where the other, similar sectors are found, so that it is logically 'prior' to them; and (3) it usually contains code which is executed when the system boots up. This sector contains a table describing at most four partitions. These areas are called *primary partitions*.

The partition table in the primary boot sector may also describe at most one *extended partition*. This is a large area of the disk, usually containing all the space which is not in any primary partition. Within this space we can set aside other areas which are called *logical partitions*, because they look almost exactly like primary partitions. In fact, the main difference between them is that we can boot from primary partitions, while we cannot boot from logical partitions. This happens because the address of a primary partition is in a fixed place, whereas the address of a secondary partition is not, so we require a more complicated process to discover it, one which is too difficult for most primary boot programs.

7.2.2. Dividing up your disk

It is a good idea to plan ahead before you start creating partitions on your disk. If you set aside a partition for some purpose, it is not easy to change its size: you must all the data from the partition, whether to floppies, to another partition, to another hard disk, or somewhere else; then you must edit the table which describes this partition, so changing its size; then you must reboot and initialise the new partition, formatting it, for example, under DOS, or running '`mkfs`' under Linux; finally you can copy all the data back. It is possible, if you have several partitions, to copy data back and forth between them while you change their sizes, but this is a bit risky and time consuming. It is better to plan ahead what you will need, since it is hard to change it afterwards.

Many people with large disks and recent versions of DOS have their entire file system on one large partition. They usually ask, 'Isn't there any way I can reformat my disk without copying everything off?' There is no way to do it using standard DOS utilities, and there is no truly safe way to do it using commercial software, because, if you make a mistake, you will lose the entire contents of your disk. If you are going to back up your disk anyway, you might as well copy the data back safely. The Linux FAQ contains references to tools and procedures which will allow you to do this, if you dare.

DOS and Linux both allow you to access several partitions on a single disk; on DOS these are treated as if they were separate disks or drives, and under Linux they are treated as different *devices*.

You can have up to 64 partitions on a single IDE disk, or up to 16 partitions on a single SCSI disk, at least as far as Linux is concerned; in practice you will rarely want so many. The maximum size of a Linux file system on a single partition depends on the type of file system you use. Minix file systems are limited to 64 megabytes. You may have all of your Linux files in a single partition, or you may have two, three, or more Linux file systems. Similarly you may have one or more DOS partitions. If you have several small partitions, you run much less risk of losing all your files if your disk gets corrupted. On the other hand, you may run out of space on a small partition more easily.

Under DOS, you must refer to each partition by a separate drive letter, but all partitions are automatically accessible. Under Linux only the root partition is automatically accessible, but once we mount another partition, it is indistinguishable from the rest of the file system. Disks are usually mounted by a command in one of the system startup files, `/etc/rc`, so you need not worry about having to do it yourself whenever you boot the system.

Linux requires at least one partition, which is the 'root' of the file system. You may prefer to have a separate partition for `/usr`, which contains most of the executable files, or for `/home`, which contains most of your private files. You may also wish to set aside a partition to use for swap space, depending on the amount of memory your PC has. You will certainly need swap space if you have less than 4 Mb of RAM and wish to compile anything substantial. You can reserve swap space in a file, but you need a partition big enough to hold it, and this will probably be less efficient than having a partition devoted to swap.

The disk space you need for Linux is discussed in Section 3.1 [Disk space required for Linux files], page 11.

Are you going to boot Linux from the hard disk, or will you boot from a floppy? Some boot programs place severe restrictions on where the boot partition can be. LILO is more relaxed about this, but does require either the Master Boot Record on your first hard disk, or the boot record on one of the first four partitions on your first hard disk.

If you have an extended partition with logical partitions in it, you can have only three primary partitions containing data.

7.2.3. The fdisk command

Every operating system, whether DOS, OS/2, or Linux, should provide its own utility for editing hard disk partition tables. At least four of these utilities have been called '`fdisk`', for 'Fixed DISK setup program', where 'fixed' means 'not removable'. I believe the first PC program named '`fdisk`' came from Microsoft in about 1985; before that time disks were too small to divide into separate sections.

Every operating system has its own peculiarities. Normally you should set up a partition for the use of one operating system by using its own '`fdisk`' program. Do not use the Linux '`fdisk`' to create partitions for DOS or for any system other than Linux; otherwise you may have problems.

An '`fdisk`' program performs two functions: it reports how the disk is configured, and it changes that configuration by adding or deleting partitions. Most '`fdisk`' programs can also change other information in partition tables.

This '`fdisk`' for Linux operates on one hard disk at a time. If you give the command

```
fdisk
```

it reports on, and is able to change, `/dev/hda`, the first hard disk. (If you have no `/dev/hda`, '`fdisk`' uses `/dev/sda` as the default device.) To look at or change the second hard disk, `/dev/hdb`, give the command

```
fdisk /dev/hdb
```

To look at or change the first SCSI disk, give the command

```
fdisk /dev/sda
```

There are some special forms of the '`fdisk`' command. One of them, suggested by Jim Winstead, simply lists all partitions on all available disks:

```
fdisk -l     (where 'l' is a letter, not the digit '1')
```

The option '`-v`' is provided to list the current version of the '`fdisk`' command. Finally, there is an option '`-s`' which is not really intended for interactive use. It causes fdisk to print the size of a partition in blocks of 1024 bytes as follows:

```
fdisk -s /dev/hda7
39934
```

Because this is intended to be used by '`mkfs`' and '`mkswap`' programs, it does not return the size of extended partitions or of partitions whose system type code is greater than 10 (hexadecimal a). If you start '`fdisk`' without using one of these special options, it responds by asking for a command:

```
Command (m for help): _
```

Each '`fdisk`' command consists of a single letter, which must be followed by `<RETURN>` before it is obeyed. Upper and lower case are not distinguished. Anything you type after the first character is ignored. Give the command '`m`', and you should see this menu:

```
Command action
   a   toggle a bootable flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   p   print the partition table
   q   quit without saving changes
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help): _
```

The simplest commands are Print, Verify, and Quit. On a small disk, the Print command might produce a display like this one:

```
Disk /dev/hda: 5 heads, 17 sectors, 977 cylinders
Units = cylinders of 85 * 512 bytes

   Device Boot  Begin   Start     End  Blocks   Id  System
/dev/hda1   *       1       1     236   10021+   1  DOS 12-bit FAT
/dev/hda2         837     837     977    5992+   5  Extended
/dev/hda3   *     237     237     836   25500   81  Linux/MINIX
/dev/hda5         837     837     936    4249+  82  Linux swap
/dev/hda6         942     942     977    1522    1  DOS 12-bit FAT
```

There are 5 partitions reported; `/dev/hda4` does not appear because it is not allocated. Partitions 1 and 3 are flagged as bootable. The size of each partition is reported in 1 kilobyte blocks; hence the primary Linux partition, partition 3, is 25 1/2 megabytes in size. The '`+`' after three of the sizes warns that these partitions contain an odd number of sectors: Linux normally allocates filespace in 1 kilobyte blocks, so the extra sector in partition 5 is wasted. Id numbers are reported in hexadecimal and explained in English.

The display/entry units may be either cylinders or sectors. The default is cylinders, but changing the units makes the print command display the following table for the system reported above:

```
Disk /dev/hda: 5 heads, 17 sectors, 977 cylinders
Units = sectors of 1 * 512 bytes

   Device Boot  Begin   Start     End  Blocks   Id  System
/dev/hda1   *       1      17   20059   10021+   1  DOS 12-bit FAT
/dev/hda2       71060   71060   83044    5992+   5  Extended
/dev/hda3   *   20060   20060   71059   25500   81  Linux/MINIX
/dev/hda5       71061   71061   79559    4249+  82  Linux swap
/dev/hda6       79985   80001   83044    1522    1  DOS 12-bit FAT
```

The start of data in both DOS partitions is 16 sectors after the beginning of the partition: this is one reason why you should use DOS's own 'FDISK' to create DOS partitions. Changing the units to sectors also affects the way in which the new partition command asks for the beginning and end of a new partition.

Warning: it is dangerous to create a new partition when the display/entry units are sectors.

The Verify command is useful because

1. It warns you if anything is wrong. **Always** give a Verify command before writing any changes to disk.
2. It reports how many unallocated sectors there are on the disk.

The Quit command is also useful. 'fdisk' does not actually change any data on your disk unless you give a Write command. If you are unhappy about any changes you may have made, give the Quit command, and your disk will remain as it was before you ran 'fdisk'. You can also interrupt 'fdisk' with 'CTRL-C'.

7.2.4. Deleting and adding partitions

Deleting a partition is simple. Give the Delete command by typing 'd'. 'fdisk' asks:

```
Partition number (1-6): _
```

Once you get this far, you must either delete a partition or interrupt the program with 'CTRL-C' (or whatever your current interrupt character is). Note:

1. You may delete a nonexistent partition. You will get a warning message.

2. You may delete an extended partition. This has the side effect of deleting all partitions greater than or equal to 5.
3. You may delete a logical partition. In that case, all partitions above it are renumbered at once. For example, if you delete partition 5, then partition 6 becomes known as partition 5, and partition 7 as partition 6.

Adding a partition is just a bit more complicated. Give the New command by typing '`n`'. '`fdisk`' allows you to

1. Create a primary partition, if there is a free slot in the primary partition table.
2. Create an extended partition if there is a free slot in the primary partition table, and if there is no extended partition.
3. Create a logical partition if an extended partition exists.

If more than one of these actions is possible, you will be asked to select Primary, Extended, or Logical, depending on what is currently permissible. Before you create a primary or an extended partition, you are asked what slot it is to have in the table (1–4).

You may not add a primary or an extended partition if the selected slot in the primary partition table is already occupied: in that case you simply return to the main menu. You are not allowed to add a new primary partition unless there are sectors available outside the extended partition. You are not allowed to add a new logical partition unless there are sectors available inside the extended partition.

If space is available, you are prompted for the first cylinder:

```
First sector (237-977): _
```

The limits are the lowest and the highest cylinders in which sectors are available in the appropriate part of the disk. Not all numbers in this range are necessarily available: they may fall inside an existing partition. If you select a cylinder which is already in use, you are told off and prompted again for the first cylinder. After selecting the first cylinder, you are prompted again:

```
Last cylinder or +size or +sizeM or +sizeK (237-836): _
```

The limits are the cylinder you have chosen as the first cylinder, and the highest cylinder which contains a legitimate upper boundary for the new partition. In other words, all numbers in the given range are legitimate, unlike those in the first range of cylinders. You may also specify the size of a partition in megabytes, kilobytes, or in the current units (cylinders or sectors). A plus sign '`+`' indicates that your answer is a size rather than a boundary, and the suffix '`m`' or '`k`' (upper or lower case) indicates that the size is not given in units of sectors or cyliners, but in megabytes or kilobytes respectively. Thus possible answers to the last cylinder request above are

700	Make cylinder 700 the last cylinder in the partition.
+300	Make cylinder 537 the last cylinder in the partition.
+15m	Make the partition at least 15 megabytes in size.
+12500k	Make the partition at least 12,500 kilobytes in size.

If you specify a size which is too large or an end which is out of range, the prompt is simply repeated.

Adding or deleting partitions has no effect unless you subsequently give the Write command. Please remember to give the Verify command first, just before giving the Write command: this is a safety precaution. After giving the Write command, you will see this message:

```
Calling BLKRRPART ioctl() to re-read partition table.
Syncing disks.
Reboot your system to ensure partition table is updated.
```

Here are some important things to note:

1. Before you reboot, you *may* run '`fdisk`' again, either to manage another disk, or to make additional changes to the same disk, or just to check that the changes have been made as you expected.
2. It is not a good idea to run any of the programs '`mkfs`', '`mkswap`', '`mount`', or '`swapon`' before you reboot. It is dangerous to run any programs, but these in particular may cause serious damage to any data on your disk, including the partition tables themselves.

7.2.5. Active flags and system types

The active flag is a bit in the partition table entry which marks a partition as bootable. This is important to some primary boot sector programs, which will not boot from an unflagged partition. Other such programs do not allow more than one partition to be flagged. Some, like LILO, ignore the flags completely. I prefer to flag all bootable partitions as active so that they stand out on the menu which '`fdisk`' lists. Fdisk prints a star after the name of a partition's device file if its active flag is set.

The Active command changes, or toggles, a partition's active flag. Give the Active command, and select a partition by number. If it was marked inactive, it will be flagged as active; if it was flagged as active, it will be marked inactive. You may set the active flag on an extended or logical partition, though the meaning of such a flag is by no means clear. This can be used to install LILO as a secondary boot loader to boot a Linux which lives on a second hard disk.

The Type command changes the ID number which describes what type a partition is. '`fdisk`' currently recognises 30 system IDs, in the sense that it prints a string for each of them, but it allows you to change any system ID to any other, with the following exceptions: you may not change any partition to or from the type Extended, and you may not change a partition whose type is Empty (0) to any other type. You may, however, change the type of any data partition to 0, which is equivalent to deleting it.

The new system ID or type code is a hexadecimal number. There are two ways of listing the numbers which '`fdisk`' recognises: use the List command, which prints the list, or use the Type command, which, when it prompts you for the code, says

```
Hex code (type L to list codes): _
```

where the upper case '`L`' is used for clarity. The codes printed are:

`1`	DOS 12-bit FAT	`9`	AIX bootable	`80`	Old MINIX	`c7`	Syrinx
`2`	XENIX root	`a`	OPUS	`81`	Linux/MINIX	`db`	CP/M
`3`	XENIX usr	`40`	Venix 80286	`82`	Linux swap	`e1`	DOS access
`4`	DOS 16-bit <32M	`51`	Novell?	`83`	Linux Ext	`e3`	DOS R/O
`5`	Extended	`52`	Microport	`93`	Amoeba	`f2`	DOS secondary
`6`	DOS 16-bit >=32M	`63`	GNU HURD	`94`	Amoeba BBT	`ff`	BBT
`7`	OS/2 HPFS	`64`	Novell	`b7`	BSDI fs		
`8`	AIX	`75`	PC/IX	`b8`	BSDI swap		

Some of these numbers are a trifle uncertain. By default (for the moment) '`fdisk`' uses a type of 81, the type used by the MINIX '`fdisk`'. When our new *ext2* file system is more stable, I shall change the default to 83. The number 81 in any case gives problems to DR-DOS 5.0 and 6.0; if you use this software, change the 81 to any other value (except 0, 5, 80, or 85; choose a value greater than '`a`' to be safest). Except for possible confusion with other systems, it does not usually matter what code you use.

The value of 82 for Linux swap partitions is my own invention, and is intended to give some recognisable distinction to the partitions when the values are displayed in hexadecimal.

New active flags and new system type codes are not written to the disk until you exit from '`fdisk`' with the Write command, as described above, in the section on deleting and adding partitions.

7.2.6. Extra commands for experts

The eXtra command '`x`' puts '`fdisk`' into 'expert' mode, in which a slightly different set of commands is available. The Active, Delete, List, New, Type, Verify, and 'eXpert' commands are not available in expert mode. The commands Write and Quit are available as in ordinary mode, the Print command is available, but produces output in a slightly different format, and of course the Menu command prints the expert menu. There are several new commands.

1. The Return command brings you back to the main menu.
2. The Extended command prints the list of table entries which point to other tables. Ordinary users do not need this information. The data is shown as it is stored. The same format is used for the expert Print command.

3. The dangerous Begin command allows you to move the start of data in a partition away from its beginning. Other systems create partitions with this format, and it is sometimes useful to be able to reproduce it.
4. The slightly dangerous Cylinders command allows you to change the available number of cylinders. For SCSI disk owners, note that we require not the actual number of physical cylinders, but the number of logical cylinders used by DOS and other operating systems.
5. The extremely dangerous Heads and Sectors commands allow you to change the number of heads and sectors. It should not be necessary to use these commands unless you have a SCSI disk, whose geometry Linux is not always able to determine. SCSI disk owners note that we need not the actual number of heads or of sectors per track, but the number believed to exist by DOS and other operating systems. **Warning**: If you set either of these numbers to a bad value, you may lose all data on your disk.

Always, after giving any of the commands Begin, Cylinder, Heads, or Sectors, you should Return to the main menu and give the Verify command.

7.2.7. Warnings for `fdisk` users

In general, you should not use this '`fdisk`' program to create partitions for other operating systems, only for Linux. Nor should you use '`fdisk`' commands from other operating systems do create partitions for Linux.

DR-DOS 5.0 and 6.0 are reported to have difficulties with partition ID codes of 80 or more. This '`fdisk`' sets the system type of new partitions to hexadecimal 81. DR-DOS seems to confuse this with hexadecimal 1, a DOS code. If you use DR-DOS, use the '`fdisk`' command '`t`' to change the system code of any Linux partitions to some number less than hexadecimal 80; I suggest 41 and 42 for the moment.

Partitioning a hard disk may destroy data which is on that disk if you are not careful. Go slowly, write down a description of what you started with, and always verify before you write.

Most operating systems and utilities expect that all partitions end at cylinder boundaries. This version of '`fdisk`' does so by default, but you can use it to create partitions which begin or end anywhere. This does not normally affect Linux, but it is very dangerous, as other operating systems (including DOS) may try to 'correct' the partition boundaries.

It is dangerous to create a new partition when the display/entry units are sectors.

The Verify command warns you if anything is wrong. **Always** give a Verify command before writing any changes to disk.

If you set the disk geometry (tracks per cylinder, or sectors per track) to an incorrect value, you may lose all data on your disk.

7.3. The *msdos* file system

The *msdos* file system provides the most convenient way of accessing DOS floppies and hard disk partitions from Linux. It allows you to mount a DOS disk or partition as part of the Linux directory tree, using the command

`mount -t msdos` *options device directory*

Here is a particular example of a '`mount`' command:

```
mount -t msdos -o ro,uid=100,umask=027,conv=text /dev/hda1 /mnt
```

There are some special options to the '`mount`' command which apply to the *msdos* file system, and which are not included in the man page for '`mount`'. Among these are the following:

`uid=nnn`
: When an *msdos* file system is mounted, its files and directories have no 'owner' as Linux understands it, so they are given, by default, the owner 'root'. If you wish to access these systems from another user name, it is convenient to pretend that another user owns everything in the mounted system. To do this, use '`uid=nnn`', where '`nnn`' is the user's ID number from `/etc/passwd`.

`umask=nnn`
: When an *msdos* file system is mounted, its files have only the 'write' permission (if they are not read-only files), and its directories have no permissions at all. Permissions are set on the files and directories using the mounter's '`umask`'. To change this, you may set the permissions by specifying an explicit '`umask`' on the '`mount`' command line.

`conv=`*conversion*
: DOS text files and Unix text files have different conventions for terminating lines. By default, *msdos* file systems are mounted in such a way that no conversion is done. This may be inconvenient if you are editing files on a DOS partition. The option '`conv=t`' or '`conv=text`' converts all line terminators when files are read from or written to the mounted DOS file system. This may lead to corruption of binary files. A safer (but still risky) option is '`conv=a`' or '`conv=auto`', which tries to guess whether a file is a text file or a binary from its extension.

7.4. Special commands for DOS floppies and partitions

Mtools is a public domain collection of programs to allow Unix systems to read, write, and manipulate files on a DOS file system. The DOS file system itself is usually a more convenient way to access DOS file systems, but mtools provides the ability to format and label DOS partitions, and they can allow non-root users to access floppy disks if you change the permissions on the '`/dev/fd*`' files.

The following DOS commands are emulated:

`mattrib`	`ATTRIB`	change DOS file attribute flags
`mcd`	`CD`	change DOS directory
`mcopy`	`COPY`	copy DOS files to/from Unix
`mdel`	`DEL/ERASE`	delete a DOS file
`mdir`	`DIR`	display a DOS directory
`mformat`	`FORMAT`	add DOS file system to a low-level format
`mlabel`	`LABEL`	make a DOS volume label
`mmd`	`MD/MKDIR`	make a DOS subdirectory
`mrd`	`RD/RMDIR`	remove a DOS subdirectory
`mread`	`COPY`	low level read (copy) a DOS file to Unix
`mren`	`REN/RENAME`	rename an existing DOS file
`mtype`	`TYPE`	display contents of a DOS file
`mwrite`	`COPY`	low level write (copy) a Unix file to DOS

These commands should not be used on mounted DOS file systems, since the kernel may not recognise that changes have been made without using the usual file system interface.

The pattern matching routine more closely resembles Unix than DOS. For example, '`*`' matches all DOS files; it is not necessary to use '`*.*`'.

The use of wildcards (or the '\' separator) requires the names to be enclosed in quotes, so that the shell does not expand them. For example,

```
mcopy "a:*.c" .
```

copies all files on the A: disk with the extension '`.C`' to the current Unix directory. On the other hand,

```
mcopy *.c a:
```

copies all files with the extension '`.c`' in the current Unix directory to the A: drive. This time you *want* the shell the expand the '`*.c`'.

The original programs have been modified so that the device configuration is read at run-time from a file `/etc/mtools`. Moreover a check is performed to detect incorrect FAT type settings.

The configuration file `/etc/mtools` defines the mapping of DOS drives to Unix devices. An entry in /etc/mtools can have any of the following formats:

drive device
drive device fat
drive device fat cylinders heads sectors
drive device fat cylinders heads sectors offset

Drive is the DOS drive letter, *device* is the name of the device on Unix, *fat* is the number of bits a FAT entry has (either 12 or 16), *cylinders*, *heads*, and *sectors* describe the disk geometry. On hard disks, the geometry parameters should all be zero. *Offset* specifies how many bytes should be skipped when accessing the device. This is necessary when using non-standard disk configurations.

The abbreviated entries are valid only on systems for which mtools can 'guess' the disk parameters by looking at the Unix device name. If this is not possible, an error message is issued.

Here is a sample configuration file:

```
# /etc/mtools - mtools device definitions

A /dev/fd0 12 80 2 15 # A: 5.25" HD
A /dev/fd0 12 40 2 9 # A: 5.25"
B /dev/fd1 12 80 2 18 # B: 3.5" HD
B /dev/fd1 12 80 2 9 # B: 3.5"
C /dev/hda1 16 0 0 0 # C: 30 MB HD partition
```

A portable entry for floppies can be created by using '`0 0 0`' for the disk geometry:

```
# /etc/mtools - portable mtools device definitions

A /dev/fd0 12 0 0 0 # any A:
B /dev/fd1 12 0 0 0 # any B:
```

This has the slight disadvantage that '`mformat`' refuses to work.

It is important to set the FAT type right because this parameter can't be computed reliably from other parameters. Mtools tries to detect possibly incorrect settings and issues an error message. If this check should be bypassed, the FAT type has to be specified as a negative number; for example:

```
D /dev/hda2 -16 0 0 0
```

Determining the correct FAT type is critical for hard disks. Writing to a hard disk with an incorrectly set FAT type will destroy vital information. If you're not sure whether you have a 12 or a 16 bit FAT, you should try reading a text file which is bigger than 8 Kb with the '`mtype`' command. You can use the commands '`mcd`' and '`mdir`' to locate such a file. The FAT type is wrong if '`mtype`' outputs data from other files after a while. The first few kilobytes are always read correctly, whether the FAT type is good or not.

7.5. Commands peculiar to Linux

There are several commands peculiar to Linux, or with features peculiar to Linux, of which you ought to be aware. Most of these commands have man pages which give useful information. After the system is installed, I suggest giving particular attention to the following commands:

```
clock(8)      fsck.minix(8) mkfs.ext2(8)    setfdprm(8)
doshell(8)    fstab(5)      mkfs.minix(8)   setserial(8)
fdformat(8)   halt(8)       mklost+found(8) shutdown(8)
frag(8)       init(8)       mkswap(8)       swapon(8)
fsck(8)       kbdrate(8)    mount(8)        update(8)
fsck.ext2(8)  mkfs(8)       rdev(8)
```

The command for reading about each of these corresponds to

```
man 5 fstab
```

where the '5' is replaced by the appropriate section number (usually '8' for these commands), and 'fstab' is replaced with the name of the command. The exception is the command 'mklost+found', whose man page can be read with the command

```
man 8 mklost+fou
```

in order to comply with the constraints imposed on the current distribution.

Appendix A. Acknowledgments

The following commands are installed from the boot floppy:

`basename`	shellutils-1.8	GNU
`bash`	bash-1.12	GNU
`brik`	brik-1.55	Rahul Dhesi
`cat`	textutils-1.4	GNU
`chgrp`	fileutils-3.5	GNU
`chmod`	fileutils-3.5	GNU
`chown`	fileutils-3.5	GNU
`clock`	utile-1.4	Charles Hedrick
`cp`	fileutils-3.5	GNU
`cpio`	cpio-2.2	GNU
`ctrlaltdel`	utile-1.4	Peter Orbaek
`cut`	textutils-1.4	GNU
`date`	shellutils-1.8	GNU
`dd`	fileutils-3.5	GNU
`df`	fileutils-3.5	GNU
`dir`	fileutils-3.5	GNU
`dirdump`	e2fsprogs-0.3	Remy Card
`doshell`	utile-1.4	Jim Wiegand
`du`	fileutils-3.5	GNU
`ed`	utilb-1.3	Brian Beattie, et al
`egrep`	grep-1.6	GNU
`env`	shellutils-1.8	GNU
`expr`	shellutils-1.8	GNU
`false`	shellutils-1.8	GNU
`fdisk`	utile-1.4	A V Le Blanc
`fgrep`	fgrep-1.1	GNU
`file`	file 1.9	Ian F Darwin
`find`	find-3.8	GNU
`frag`	utile-1.4	Werner Almesberger
`free`	procps-0.7	Brian Edmonds
`fsck`	fsutil-1.6	David Engel
`fsck.ext2`	e2fsprogs-0.3	Remy Card
`fsck.minix`	utile-1.4	Linus Torvalds
`fuser`	procps-0.7	Werner Almesberger
`getty`	utile-1.4	W Z Venema
`grep`	grep-1.6	GNU
`gunzip`	gzip-1.0.7	GNU
`gzip`	gzip-1.0.7	GNU
`halt`	svinit 2.0	Miquel van Smoorenburg
`head`	textutils-1.4	GNU
`hostname`	utilb-1.3	Peter Orbaek
`Image`	linux-0.99p8a	Linus Torvalds
`init`	svinit 2.0	Miquel van Smoorenburg
`install`	fileutils-3.5	GNU

`kill`	utilb-1.3	Peter MacDonald
`killall`	procps-0.7	Werner Almesberger
`last`	svinit 2.0	Miquel van Smoorenburg
`ldd`	utilb-1.3	David Engel
`less`	less-1.77	GNU
`lesskey`	less-1.77	GNU
`ll`	fileutils-3.5	GNU
`ln`	fileutils-3.5	GNU
`login`	utilb-1.3	BSD
`ls`	fileutils-3.5	GNU
`lsf`	fileutils-3.5	GNU
`mesg`	svinit 2.0	Miquel van Smoorenburg
`mkdir`	fileutils-3.5	GNU
`mkfifo`	fileutils-3.5	GNU
`mkfs`	fsutil-1.6	David Engel
`mkfs.ext2`	e2fsprogs-0.3	Remy Card
`mkfs.minix`	utile-1.4	Linus Torvalds
`mklost+found`	e2fsprogs-0.3	Remy Card
`mknod`	fileutils-3.5	GNU
`mkswap`	utile-1.4	Linus Torvalds
`more`	utilb-1.3	BSD
`mount`	mount-0.99.6	Doug Quale
`mv`	fileutils-3.5	GNU
`newgrp`	-	Michael Haardt
`passwd`	utila-1.5	Peter Orbaek
`printenv`	shellutils-1.8	GNU
`ps`	procps-0.7	Branko Lankester
`pstree`	procps-0.7	Werner Almesberger
`psupdate`	procps-0.7	Branko Lankester
`ramsize`	utile-1.4	Werner Almesberger
`rdev`	utile-1.4	Werner Almesberger
`reboot`	svinit 2.0	Miquel van Smoorenburg
`rm`	fileutils-3.5	GNU
`rmdir`	fileutils-3.5	GNU
`sed`	sed-1.13	GNU
`setfdprm`	utile-1.4	Werner Almesberger
`setserial`	utile-1.4	Rick Sladkey
`setterm`	utilb-1.3	Gordon Irlam
`sh`	bash-1.12	GNU
`shutdown`	svinit 2.0	Miquel van Smoorenburg
`sort`	textutils-1.4	GNU
`strings`	utila-1.5	BSD
`stty`	shellutils-1.8	GNU
`su`	shellutils-1.8	GNU
`swapoff`	mount-0.99.6	Doug Quale
`swapon`	mount-0.99.6	Doug Quale
`sync`	utile-1.4	Linus Torvalds
`tail`	textutils-1.4	GNU

telinit	svinit 2.0	Miquel van Smoorenburg
test	shellutils-1.8	GNU
tload	procps-0.7	Branko Lankester
top	procps-0.7	Branko Lankester, Roger Binns
touch	fileutils-3.5	GNU
true	shellutils-1.8	GNU
umount	mount-0.99.6	Doug Quale
update	utile-1.4	Linus Torvalds
uptime	procps-0.7	Michael K Johnson
vdir	fileutils-3.5	GNU
vidmode	utile-1.4	Werner Almesberger
w	procps-0.7	Larry Greenfield
wall	svinit 2.0	Miquel van Smoorenburg
who	shellutils-1.8	GNU
write	utilb-1.3	BSD
xargs	find-3.8	GNU
zcat	gzip-1.0.7	GNU

The following commands are contained in the *base* package:

apropos	man-1.1	John Eaton
banner	utila-1.5	BSD
cal	utila-1.5	BSD
chsh	utila-1.5	Peter Orbaek
cksum	textutils-1.4	GNU
clear	-	A V Le Blanc
cmp	diff-2.2	GNU
col	utila-1.5	BSD
colcrt	utila-1.5	BSD
colrm	utila-1.5	BSD
column	utila-1.5	BSD
comm	textutils-1.4	GNU
compress	compress-4.0.1	Joseph M Orost
csplit	textutils-1.4	GNU
ctags	elvis-1.7	Steve Kirkendall
diff	diff-2.2	GNU
diff3	diff-2.2	GNU
dirname	shellutils-1.8	GNU
echo	shellutils-1.8	GNU
elvis	elvis-1.7	Steve Kirkendall
elvprsv	elvis-1.7	Steve Kirkendall
elvrec	elvis-1.7	Steve Kirkendall
ex	elvis-1.7	Steve Kirkendall
expand	textutils-1.4	GNU
fdformat	utile-1.4	Werner Almesberger
fmt	elvis-1.7	Steve Kirkendall
fold	textutils-1.4	GNU
groups	shellutils-1.8	GNU

gzexe	gzip-1.0.7	GNU
hexdump	utila-1.5	BSD
id	shellutils-1.8	GNU
joe	joe-0.1.5	Joseph H Allen
join	textutils-1.4	GNU
kbdrate	utile-1.4	Rickard E Faith
lilo	lilo9	Werner Almesberger
locate	find-3.8	GNU
logger	utila-1.5	BSD
logname	shellutils-1.8	GNU
look	utila-1.5	Rickard E Faith
makehole	utile-1.4	H J Lu
makewhatis	-	Rickard E Faith
man	man-1.1	John Eaton
manpath	man-1.1	John Eaton
mattrib	mtools-2.0	Emmet P Gray
mcd	mtools-2.0	Emmet P Gray
mcopy	mtools-2.0	Emmet P Gray
mdel	mtools-2.0	Emmet P Gray
mdir	mtools-2.0	Emmet P Gray
mformat	mtools-2.0	Emmet P Gray
mkmanifest	mtools-2.0	Emmet P Gray
mlabel	mtools-2.0	Emmet P Gray
mmd	mtools-2.0	Emmet P Gray
mrd	mtools-2.0	Emmet P Gray
mread	mtools-2.0	Emmet P Gray
mren	mtools-2.0	Emmet P Gray
mtype	mtools-2.0	Emmet P Gray
mwrite	mtools-2.0	Emmet P Gray
nice	shellutils-1.8	GNU
nl	textutils-1.4	GNU
nohup	shellutils-1.8	GNU
od	textutils-1.4	GNU
paste	textutils-1.4	GNU
patch	patch-2.0.12g8	Larry Wall
pathchk	shellutils-1.8	GNU
pr	textutils-1.4	GNU
printf	shellutils-1.8	GNU
ref	elvis-1.7	Steve Kirkendall
script	utila-1.5	BSD
sdiff	diff-2.2	GNU
sleep	shellutils-1.8	GNU
split	textutils-1.4	GNU
sum	textutils-1.4	GNU
syslogd	utile-1.4	BSD
tac	textutils-1.4	GNU
tee	shellutils-1.8	GNU
time	time-1.4	GNU

`tr`	textutils-1.4	GNU
`tty`	shellutils-1.8	GNU
`ul`	utila-1.5	BSD
`uname`	shellutils-1.8	GNU
`uncompress`	compress-4.0.1	Joseph M Orost
`unexpand`	textutils-1.4	GNU
`uniq`	textutils-1.4	GNU
`updatedb`	find-3.8	GNU
`users`	-	?
`uudecode`	utila-1.5	BSD
`uuencode`	utila-1.5	BSD
`vi`	elvis-1.7	Steve Kirkendall
`virec`	elvis-1.7	Steve Kirkendall
`wc`	textutils-1.4	GNU
`whatis`	man-1.1	John Eaton
`whereis`	utila-1.5	BSD
`whoami`	shellutils-1.8	GNU
`yes`	shellutils-1.8	GNU
`zcmp`	gzip-1.0.7	GNU
`zdiff`	gzip-1.0.7	GNU
`zforce`	gzip-1.0.7	GNU
`zmore`	gzip-1.0.7	GNU
`znew`	gzip-1.0.7	GNU

The following commands are contained in the *groff* package:

`addftinfo`	groff-1.07	GNU
`afmtodit`	groff-1.07	GNU
`geqn`	groff-1.07	GNU
`gindxbib`	groff-1.07	GNU
`glookbib`	groff-1.07	GNU
`gnroff`	groff-1.07	GNU
`gpic`	groff-1.07	GNU
`grefer`	groff-1.07	GNU
`grodvi`	groff-1.07	GNU
`groff`	groff-1.07	GNU
`grog`	groff-1.07	GNU
`grops`	groff-1.07	GNU
`grotty`	groff-1.07	GNU
`gsoelim`	groff-1.07	GNU
`gtbl`	groff-1.07	GNU
`gtroff`	groff-1.07	GNU
`lkbib`	groff-1.07	GNU
`pfbtops`	groff-1.07	GNU
`psbb`	groff-1.07	GNU
`tfmtodit`	groff-1.07	GNU

The following commands are contained in the *tcpip* package:

`config`	netsrc21m	?
`finger`	netsrc21m	BSD
`fingerd`	netsrc21m	BSD
`ftp`	netsrc21m	BSD
`ftpd`	netsrc21m	BSD
`inetd`	netsrc21m	BSD
`myaddr`	netsrc21m	A V Le Blanc
`named`	netsrc21m	BSD
`nslookup`	netsrc21m	BSD
`nsquery`	netsrc21m	BSD
`nstest`	netsrc21m	BSD
`ntalkd`	netsrc21m	BSD
`ping`	netsrc21m	BSD
`rcp`	netsrc21m	BSD
`rlogin`	netsrc21m	BSD
`rlogind`	netsrc21m	BSD
`rsh`	netsrc21m	BSD
`rshd`	netsrc21m	BSD
`talk`	netsrc21m	BSD
`telnet`	netsrc21m	David Borman
`telnetd`	netsrc21m	David Borman
`tftp`	netsrc21m	BSD
`tftpd`	netsrc21m	BSD
`wdsetup`	netsrc21m	Gregg Weber

The following commands are contained in other packages, as indicated:

`ar`	*gcc*	binutils-1.9l	GNU
`as`	*gcc*	binutils-1.9l	GNU
`as86`	*gcc*	bin86	Bruce Evans
`awk`	*gawk*	gawk-2.14	GNU
`bison`	*bison*	bison-1.21	GNU
`c++`	*gpp*	gcc-2.3.3	GNU
`cc`	*gcc*	gcc-2.3.3	GNU
`cpp`	*gcc*	gcc-2.3.3	GNU
`ctags`	*emacs*	emacs-18.59	R M Stallman
`emacs`	*emacs*	emacs-18.59	R M Stallman
`emacsclient`	*emacs*	emacs-18.59	R M Stallman
`env`	*emacs*	emacs-18.59	R M Stallman
`etags`	*emacs*	emacs-18.59	R M Stallman
`flex`	*flex*	flex-2.3.8	Vern Paxson
`g++`	*gpp*	gcc-2.3.3	GNU
`gawk`	*gawk*	gawk-2.14	GNU
`gcc`	*gcc*	gcc-2.3.3	GNU
`gdb`	*gdb*	gdb-4.8	GNU
`genclass`	*gpp*	gcc-2.3.3	GNU

`gprof`	*gprof*	binutils-1.9l	GNU
`info`	*info*	texinfo-2.16	GNU
`kermit`	*kermit*	ckermit-5a-188	Frank da Cruz
`ld`	*gcc*	binutils-1.9l	GNU
`ld86`	*gcc*	bin86	Bruce Evans
`make`	*gcc*	make-3.64	GNU
`makeinfo`	*info*	texinfo-2.16	GNU
`movemail`	*emacs*	emacs-18.59	R M Stallman
`nm`	*gcc*	binutils-1.9l	GNU
`objdump`	*gcc*	binutils-1.9l	GNU
`ranlib`	*gcc*	binutils-1.9l	GNU
`server`	*emacs*	emacs-18.59	R M Stallman
`size`	*gcc*	binutils-1.9l	GNU
`strip`	*gcc*	binutils-1.9l	GNU
`tar`	*tar*	tar-1.11.2	GNU

Concept Index

File Index

U

W

X

Program Index

Linux programs appear in lower case, and DOS programs appear in upper case.

A

B

C

D

E

F

U

V

W

X

Y

Z

Table of Contents