MCC Interim Linux

The MCC Distribution of the Linux Operating System

Version 1.0+

April 1994

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 looks 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 performs slowly without at least 8 Mb.
- Linux requires an ISA or EISA bus. It runs on systems with a local bus architecture, such as VESA. At present it does not support the MCA bus architecture; this means that is does not run on PS/2 machines.
- Linux supports AT standard (16-bit) hard disk controllers, including IDE, ESDI, MFM, and RLL controllers. It supports generic XT controllers (8 bit hard disk controllers with MFM or RLL). It also supports many SCSI controllers and CD ROMs, including Adaptec 1542 (but not 1522), Adaptec 1740 in extended mode (not in 1542 compatible mode), Seagate ST-01 and ST-02, NCR-5380, PAS-16, Trantor T128, T128F, and T228, Future Domain 1680, TMC-850, any board based on the TMC-950 chip, TMC-1660 and 1680, Ultrastor 14F, 24F, and 34F, and WD7000. Clones of these SCSI controller cards should also work with Linux.
- 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 60 Mb or more. See Chapter 3 [Preparing to Install Linux], page 15.
- Linux supports several Ethernet cards. The 'official' kernel currently contains code for NE1000, NE2000, NE2100, SMC Ultra, 3c501, 3c503, 3c509, 3c579, HP PCLAN, AT1500, AT1700, DEPCA, D-Link DE600, AT-LAN-TEC/RealTek, and Western Digital 8003 and 8013 cards. Patches for other cards are available, and will eventually be included in the 'official' kernel. The most common cards in Manchester are RM Ethernet cards. Some of these can be modified to work as NE2000 cards: AT cards with serial numbers above 20000. We have been unable to get RM Ethernet cards to work reliably with Linux, and Research Machines have been unwilling to supply information which could help us to get their cards working, or only if we sign a non-disclosure agreement.
- 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, Mach32, Mach8, and monochrome video boards such as Hercules.

Detailed information about currently supported hardware is published in the Linux hardware HOWTO. See Section 7.3 [The Linux HOWTO documents], page 66.

Among the features of Linux are the following:

- It is a multi-tasking system: several programs can run at the same time. You can run several programs simultaneously on several virtual consoles, in several windows, or by logging in remotely.
- 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. Even a data page is shared until one program writes to it. This increases speed and decreases memory use.
- Programs access virtual memory which is paged to disk. You may allocate up to 16 swap areas, each up to 16 Mb in size. This memory forms a pool which all running processes share.
- 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. The source code for every single binary and library is available with the MCC distribution. See Section 2.1 [Getting MCC Interim Linux by anonymous ftp], page 7. Moreover, the optional *patches* package contains all patches applied to these sources to make the MCC distribution.

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 6.1 [Caring for your Linux system], page 49.

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 did not contain an executable shell. With release 1.0+, the minimum base system no longer fits comfortably on a single disk, so we return to having two disk images, neither of which contains an executable shell; once more you can install the MCC version from a ramdisk, though this is only one of the available options.

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. More recently the Slackware and Debian versions have appeared, as well as a number of commercial releases. 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. 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 binaries which we do not recompile are the C and C++ 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. For information about installing XFree86 release 2.1, see Section 5.6 [Installing the XFree86 software], page 47. The *emacs* and *info* packages were added to MCC Interim Linux only because we have been using them for our courses, and because of the value of the information included in the info files. See Section 7.1 [Documentation available on line], page 65. People who want a distribution which contains nearly everything should get the Debian version, which is 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.

The MCC distribution allows you to upgrade an existing system, as well as to install a complete new system. The main limitation is 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 after you recompile the kernel. As of release 1.0+, *minix* file systems must support 30-character filenames, or the MCC distribution will fail to install properly. 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`, `home/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 5 [Tailoring MCC Interim Linux to Your Taste], page 41.

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 69.) 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 installed as `/usr/lib/COPYING`.

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. Even after the release and the documentation have been frozen, the file `Bugs+Warnings` is updated. Please read it carefully.

I have fixed many bugs, but I am certain I have created others. I would appreciate any bug reports or fixes. Please send them by E-mail to me at `LeBlanc@mcc.ac.uk`.

1. This distribution, even when it is used to upgrade an existing system, **must** be installed from the boot and root disks. If you try to install MCC packages without using the boot and root disks, they will probably not work properly.
2. As of release 1.0+, MCC Interim Linux does not install correctly on a *minix* file system unless it accepts filenames of up to 30 characters in length. The '`mkfs`' command on the root disk does not allow you to create a *minix* file system with the 14-character limit on file name length.
3. The Linux NFS implementation is limited by the size of the memory which the kernel can allocate for buffers. Heavy reading or writing of files in an NFS-mounted file system may cause the kernel to hang unless you specify the options '`rsize=1024,wsize=1024`' either on the '`mount`' command or in the entry in `/etc/fstab`.

4. Older versions of the SysVinit package allowed root to enter single user mode without a password, but required the root password when the system was booted in single user mode. In attempting to restore this behaviour, I have introduced a new bug. After you boot the system in single user mode, you can change to another run level, but if you as root change back to single user mode without rebooting, you must type in the root password again.
5. The '`more`' command does not work properly under X.
6. The root and boot disks, as well as the installed system, contain device files for eight partitions on each disk. If you need more partitions, you must use '`MAKEDEV`' or '`mknod`' to create additional device files in the installed `/dev` directory.
7. It is not our policy to make extensive modifications to the Linux kernel. Bugs in the current kernel may be fixed in later versions of the kernel.
8. After the package is released, small bugs are often discovered. The Bugs+Warnings file may be consulted for further information.

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.

2.1. Getting MCC Interim Linux by anonymous ftp

Anonymous ftp is a service which allows people with TCP/IP software and Ethernet access to retrieve files from other machines, called servers. You do not need a user name on the ftp servers. The required software is installed on most Unix machines. Several versions of the software are available for machines running MS-DOS. A full tutorial in the use of this software is beyond the scope of this document, but the following sample session should convey a general idea of how the software can be used. A longer introduction to anonymous ftp can be found in *Linux Installation and Getting Started*; for further information, see Section 7.5 [Other information about Linux], page 67.

```
$ ftp ftp.mcc.ac.uk
Connected to cfs2.mcc.ac.uk.
220 cfs2 FTP server (Version wu-2.1c(8) date) ready.
Name (myname:my.address): _
```

On some machines it is necessary to give the ftp command or its equivalent without an argument, then to use the 'open' command to connect to the remote site. On some machines it is necessary to give an IP address instead of the name of the remote machine; the IP address for `ftp.mcc.ac.uk` is 130.88.203.12.

```
Name (myname:my.address): ftp
331 Guest login ok, send your complete e-mail address as password.
Password (linux:ftp): Myname@my.address
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> _
```

The user name '`ftp`' usually gives access to the anonymous ftp service. On some machines it is necessary to give the user name '`anonymous`'. The password is not in fact a password at all, but your identification. It is customary — and more courteous — to give your own E-mail address as the password. Sometimes this allows us to contact people who have inadvertently copied software which contains dangerous bugs. Some machines will not let you have access to their anonymous ftp service unless you give a valid E-mail address as the password.

```
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 4
drwx--x--x   2 admin    sys          2048 Feb 26 19:34 bin
dr-x--x--x   2 admin    sys          2048 Mar 19 17:57 etc
dr-x--x--x   2 admin    sys          2048 Apr  5 13:50 logs
```

```
drwx--x--x   2 admin    sys          2048 Mar 18 15:16 mirror
dr-xr-xr-x   3 admin    sys          2048 Mar  2 11:29 pub
226 Transfer complete.
ftp> _
```

The '`dir`' command gives you a listing of the remote directory.

```
ftp> cd pub/linux
250 CWD command successful.
ftp> cd mcc-interim
250 CWD command successful.
ftp> cd 1.0+
250 CWD command successful.
ftp> _
```

Note that you may move through the directory tree one step at a time, or many steps at once.

```
226 Transfer complete.
ftp> ascii
200 Type set to A.
ftp> get README.lilo
200 PORT command successful.
150 Opening ASCII mode data connection for README.lilo (45088 bytes).
226 Transfer complete.
46057 bytes received in 0.259 secs (1.7e+02 Kbytes/sec)
```

The '`ascii`' command ensures that subsequent transfers are made in text mode. This is a good idea unless you are certain that the remote machine uses the same format and character set for text files that yours does. For example, MS-DOS and Unix do not use the same format for text files.

The '`get`' command copies a remote file to your machine. Some ftp programs allow you to change the name of the file when it is transferred. Note also that the names of some files are changed automatically if you transfer them to MS-DOS systems; for example, `README.lilo` becomes `README.LIL`.

```
ftp> cd images
250 CWD command successful.
ftp> binary
200 Type set to I.
ftp> mget r*
mget root.gz? y
200 PORT command successful.
150 Opening BINARY mode data connection for root.gz (844253 bytes).
226 Transfer complete.
844253 bytes received in 1.75 secs (4.7e+02 Kbytes/sec)
ftp> quit
221 Goodbye.
$ _
```

The '`binary`' command ensures that subsequent transfers are made in binary mode. You should use this mode for all MCC packages and compressed disk images, as well as for all MS-DOS executable files.

The '`mget`' command gets all files whose name matches the pattern you give. '`mget *`' gets all files in the remote directory. Some ftp servers, such as the one at `ftp.mcc.ac.uk`, allow you to get a whole directory tree as a single tar file, or as a compressed or gzipped tar file. For example, to retrieve the entire MCC 1.0+ distribution (including source files) as a tar file, change to the directory `/pub/linux/mcc-interim`, and give the command '`get 1.0+.tar`'. Note that the resulting tar file is about 70 Mb in size.

The '`hash`' command prints a `#` for every kilobyte transferred. This may be useful if your connection does not allow fast transfers.

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), `1.0+` (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`, from `tsx-11.mit.edu [18.172.1.2]` 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 `1.0+` 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 two boot disks and the root or base disk. The file `nocdboot.gz` is the normal boot disk, while `cdrmboot.gz` is a special boot disk, whose kernel contains drivers for the supported CD ROM disks, but which may cause problems if you have other hardware. `root.gz` is the root or base disk. The suffix '`.gz`' indicates that the files have been gzipped, and must be uncompressed with '`gunzip`' or with the command '`gzip -d`'. The files `nocdboot.cnf` and `cdrmboot.cnf` are copies of the `config.in` files used to compile the kernels on these disks.

'`gzip`' is the GNU compression utility. A version of this utility which runs under MS-DOS can be found in the `dos-utils` directory.

`packages` This contains the packages which are optional, but important, parts of the MCC Interim distribution.

`extra_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 one of them, `wd.tgz`, contains support only for the Western Digital 80x3 Ethernet cards. All four MCC kernels contain maths coprocessor emulation code, and support for IDE hard disks, parallel printers, selection (cut and paste between virtual consoles), ELF and COFF binary support, and support for the *minix*, *ext2*,

msdos, NFS, and *proc* file systems. The files `ipide.cnf` and `wd.cnf` are copies of the `config.in` files used to compile these kernels.

cdrmboot All CD ROM, all SCSI, XD, all Ethernet cards.

nocdboot No CD ROM, all SCSI, XD, all Ethernet cards.

ipide No CD ROM, no SCSI, no XD, all Ethernet cards.

wd No CD ROM, no SCSI, no XD, WD 80x3 Ethernet cards only.

`extra_packages`

This contains packages which have a secondary status, mainly because they are not needed to recompile other packages.

emacs,

elisp A cut-down version of the '`emacs`' editor, which we use in our courses at the Computer Centre. A more complete version can be obtained by installing the *emacsxtr* package in addition to these two. Note that the *emacsxtr* package is too large to fit on a floppy disk. A version of '`emacs`' which is compiled with support for X windows is available by anonymous ftp from the `contributions` directory.

extrainf Info files which I did not wish to include in the main distribution; these describe the GNU assembler '`gas`', '`bison`', '`flex`', texinfo, '`gcc`' and how to port it, '`cpp`', some of the odder parts of '`emacs`', and other subjects.

extralib A number of libraries which are of less general interest than those contained in the *gccb* package: `bfd`, `dbm`, `gmon`, `iberty`, `mcheck`, `mmalloc`, `opcodes`, and `readline`.

gprof The GNU profiler '`gprof`', together with its support files.

kermit The '`kermit`' terminal emulator and file transfer utility.

lp The Berkeley lineprinter program '`lpr`', together with associated commands and daemons.

mail The '`elm`' mail reader and the '`smail`' mail delivery system, together with their support files.

manpages Unformatted manual pages. The formatted pages are all that most people need, and they are already contained in the other packages.

patches The patches which, when applied to the corresponding source files, produce the sources compiled in the current release.

timezone The time zone data files, which are produced from the `utilx15` source file.

words The file `/usr/dict/words`.

`source_files`

This contains the source files which were used to compile all binaries in the current distribution.

`contributions`

This contains MCC-style packages and source files for programs not distributed as part of MCC Interim Linux. The file `00-contents` describes what is there. Currently it contains NFS server software, cron, a version of the emacs binary that supports X windows, the network time software package `xntp`, and the typesetting utility TEX.

After you decide what you need, you may transfer those files in binary mode to any machine which supports ftp. The root and boot disks 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 an unformatted 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 root.gz
dd if=root of=/dev/fd0 bs=18432         (for 3.5 inch disks)
dd if=root of=/dev/fd0 bs=15360         (for 5.25 inch disks)
```

Then place another unformatted floppy in drive 0 and repeat this process with `nocdboot.gz`. (We recommend using unformatted floppies, since older floppies often have bad blocks, which make the critical root and boot floppies unreliable.)

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 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 images and copy then to a disk. The commands '`GZIPxxx.EXE`', '`RAWRITE.EXE`', and '`RAWRITE3.COM`' can be found at `ftp.mcc.ac.uk` in the directory `/pub/linux/mcc-interim/dos-utils`. Transfer '`GZIPxxx.EXE`' and one of the '`RAWRITE`' commands in binary mode. (The version number *xxx* is currently 124.) The command '`GZIPxxx.EXE`' is a self-unpacking archive, which creates '`GZIP.EXE`' and a documentation file. Documentation for the two '`RAWRITE`' commands can be found in the `dos-utils` directory with the commands. Remember to transfer documentation files in text mode. After unpacking '`GZIP.EXE`', you should give the following commands:

```
FORMAT A:
GZIP -D ROOT.GZ
RAWRITE
Enter source file name: ROOT
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`' program.

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 of the packages and extra packages, they will fit on floppies as follows:

3.5 inch high density floppies:

disk 1	*elisp*
disk 2	*bison*, *gcca*, *groff*
disk 3	*baseman*, *flex*, *gccb*, *words*
disk 4	*info*, *manpages*
disk 5	*gpp*, *mail*, *patches*
disk 6	*extrainf*, *gdb*, *kermit*
disk 7	*emacs*, *extralib*, *gawk*, *progman*
disk 8	*gprof*, *linux*, *lp*, *timezone*

In this case, disk 1 also has room for the two extra kernel packages.

5.25 inch high density floppies:

disk 1	*info*
disk 2	*gcca*, *mail*
disk 3	*baseman*, *manpages*
disk 4	*bison*, *gawk*, *groff*, *lp*, *timezone*
disk 5	*gccb*, *gdb*
disk 6	*elisp*, *progman*, *words*
disk 7	*extralib*, *flex*, *gpp*
disk 8	*extrainf*, *gprof*, *patches*
disk 9	*emacs*, *kermit*
disk 10	*linux*

In this case as well, disk 1 also has room for the two extra kernel packages.

The MCC installation allows you to create your own packages, which you can install along with ours. They must be gzipped tar files with the extension '`.tgz`'. Each package is unzipped and untarred in the root directory, but should contain only files with relative path names; e.g., not '`/usr/bin/cmd`' but '`usr/bin/cmd`'. If it contains files `install.setup` and `install.clean`, these are executed as shell scripts 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. This '`.crc`' file is then saved in `/tmp`, where you may delete it, or move it somewhere safe if you wish to keep it.

You may omit any packages you wish. Obviously, you cannot recompile the kernel unless you also install *gcca* and *gccb*. *Bison*, *flex*, *gdb*, *gpp*, and *gprof* will not work properly unless *gcca* and *gccb* are also installed.

2.2. Getting MCC Interim Linux from the MCC shop

MCC Interim Linux is available from the MCC shop. You can buy a set of diskettes containing the boot and root disks, all the packages, and the two optional kernels with no SCSI support. See Section 2.1 [Getting MCC Interim Linux by anonymous ftp], page 7, where these optional kernels are described. At present only 3.5 inch diskettes are available on demand, since there has been little call for the 5.25 inch disks.

You can also obtain a copy of this document. Because of changes in University funding policy, there is now a small charge for almost all documents, including this one.

2.3. Installing MCC Interim packages using NFS

It is 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 Ethernet card. You must already have networking software installed and configured. In the present release of MCC Interim Linux, the networking software can be installed and configured from the boot and root disks.

The mounting of remote file systems varies from one site to another. The following commands can be used in Manchester to make MCC packages available using NFS and to install them. After installing or upgrading from the boot and root disks, as described in Chapter 4 [Installing or Updating MCC Interim Linux], page 19, give the following commands:

```
mkdir /afs /packages
mount -t nfs -o rsize=1024,wsize=1024 zeus.mcc.ac.uk:/afs /afs
cd /packages
ln -s /afs/mcc/ftp/pub/linux/mcc-interim/1.0+/[pe]*/* .
```

This places symbolic links to all MCC Interim packages, extra packages, and extra kernels in the directory `/packages`. Use the command '`ls`' to list the available packages, and use the command '`rm`' to delete the links to any packages you do not wish to install.

Note: Do not install the extra kernel packages *ipide* or *wd* if you have SCSI disks, as these kernels do not contain support for them.

Give the following command:

/tmp/bootinstall

```
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

? 6
Do you wish to have a prompt before each package (Y or N): n

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

In what directory are the MCC packages to be found?
/packages
```

At this point the packages you have selected will be installed. After you have completed the installation, you may give these commands to 'clean up':

```
umount /afs
rm -rf /afs /packages
```

If MCC Interim packages are available from other sites using NFS, you should check with the local administrator to find what commands you should use to mount the appropriate file system, and to ensure that you have the appropriate permission to mount it.

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 the boot and root disks. Your 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 6.2.2 [Dividing up your disk], page 51.

3.1. Disk space required for Linux files

People often ask how much disk space Linux requires. It is difficult to give a simple answer to this question, since both file systems and swap space have different sizes, depending both on your hardware configuration and on the packages you select for your system.

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. The exact size depends on the type of file system on which the package is installed; packages usually take up less space on extended(2) file systems. 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`' and changed by the command '`tune2fs`'. (The MCC distribution does not allow you to change the 5% default before installation.) The two *base* packages must be installed from the boot and root floppies before any other packages are installed.

Package	Size in Kb	Contents
base1	2032	basic binaries and text (on root floppy)
base2	1731	basic binaries and text (on boot floppy)
tcpip	624	commands and daemons for networking (on boot floppy)
baseman	599	man pages for base package
bison	94	the GNU `yacc`-compatible parser generator
elisp	2664	lisp support files for the `emacs` editor
emacs	3039	the `emacs` editor, without lisp files
extrainf	3257	info files for `gas`, `gcc`, odd `emacs` modes
extralib	499	libraries not included in `gccb` package
flex	244	the GNU fast lexical analyser generator
gawk	130	the GNU version of `awk`
gcca	1817	the GNU C compiler, binaries and include files
gccb	2279	the GNU C libraries and code generator
gdb	693	the GNU debugger
gpp	2624	the GNU C++ compiler `g++`
gprof	522	the GNU profiler for use with `gcc` and `g++`

groff	2666	the GNU clone of the `nroff/troff` text formatter
info	2008	the GNU utility `info`, plus selected info files
kermit	378	the Kermit communications utility
linux	4685	the kernel source for Linux 0.99 patch level 10
lp	112	programs and daemons for local and remote printing
mail	994	programs for sending, receiving, and reading E-mail
manpages	3025	unformatted manual pages
patches	180	the patches used to compile this release
progman	422	manual pages for libraries and kernel
timezone	165	data files and programs concerning time zones
words	402	a list of words for the `look` command
Total	37885	

Note that this does not include the *emacsxtr* package, which requires another 8827 Kb, bringing the total to 46712 Kb if it is installed.

The amount of disk space you require depends on what software you intend to install. For all MCC packages (not including extras), you need about 23 Mb, while the extras require another 15 Mb. For the X window system (which is not included in the MCC distribution), you need a further 20 Mb. See Section 5.6 [Installing the XFree86 software], page 47. TEX (which is also not included in the MCC distribution), requires at least 12 Mb more. The source files for all MCC packages (which are also not part of the distribution) require a further 36 Mb. Additionally, you must consider the space you need for your work, perhaps an additional 5 or 10 Mb for small projects, 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 each SCSI disk, and up to 64 partitions on each 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 personal 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` or `/var`. 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.

Excluding `/usr` and `/var`, the MCC distribution takes less than 2 Mb of disk space on the root partition, though you may wish to leave more for files in `/tmp`. If you have `/var` as a separate partition, you should probably give it at least 5 Mb, depending on how big your logfiles and print queues will be. Of course, if you install news reading software, the `/var` partition may need to be much bigger. The `/usr/lib` tree requires nearly 18 Mb, and `/usr/src` at least 5 Mb or more. The remainder of the `/usr` tree takes up about 20 Mb. See Section 6.2.2 [Dividing up your disk], page 51.

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 and optimises a very large function, 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. Some applications running under X windows may require even more swap space.

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`. The online manual page for '`mkswap`' contains instructions for making new swap partitions and swap files.

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 suitable MCC boot and root disks. See Section 2.1 [Getting MCC Interim Linux by anonymous ftp], page 7, where the preparation of boot and root disks is described.

4.1. Overview of an MCC installation or upgrade

In this section I summarise what you should do to install or upgrade 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 15.
- Back up your hard disk.
- Get the boot and root images, packages, and DOS programs by using anonymous ftp in binary mode. See Section 2.1 [Getting MCC Interim Linux by anonymous ftp], page 7.
- Format some high density floppies, and copy the packages you select to them, leaving two floppies for the boot and root images. Have a spare formatted floppy which you will use to test your LILO installation before overwriting your master boot record.
- Unpack '`GZIPxxx.EXE`'.
- Use the new '`GZIP.EXE`' to unpack the boot and root images.
- Use '`RAWRITE`' to copy the boot and root images to high density floppies. See Section 2.1 [Getting MCC Interim Linux by anonymous ftp], page 7.
- Boot from the MCC boot disk. Give the '`ro`' command at the LILO prompt to boot in read-only mode. At the prompt, place the MCC root disk in the drive and press `<RETURN>`. See Section 4.2 [Booting from the MCC boot disk], page 20.
- Use '`fdisk`' to edit your partition table. See Section 6.2 ['`fdisk`': the Linux partition table editor], page 50.
- Use '`mkswap`' to set up a swap partition. See Section 4.4.1 [Initialising a swap partition], page 25.
- Use '`mkfs`' to create file system partitions. See Section 4.4.3 [Creating a file system], page 26.
- Install the *base1* package from the MCC boot disk. See Section 4.5.1 [Installing base packages using the '`ro`' option], page 28.
- After installing *base1*, reboot by answering '`y`'. Write down the instruction '`ro root=xxx`' which is printed at this point, then press `<RETURN>`.
- Reboot from the MCC boot disk, answering the LILO prompt with the command '`ro root=xxx`', according to the instructions you noted during the previous step. Leave the boot disk in the drive while the *base2* package is installed.
- Change your default keyboard map from uk.map to your national keyboard when you are asked to do so, unless you have a UK keyboard.
- Install and configure networking software when you are asked to do so.

- Remove the boot disk from the floppy drive, and use LILO to install boot code on a floppy by answering the questions which appear at this time.
- Install the packages you select from floppy disks. See Section 4.6 [Installing the MCC Interim packages], page 32.
- Reboot from your new boot floppy.
- Log in as root, and use the '`lilo`' command to install boot code on your hard disk.

The following sections explain this process in greater detail and consider some of the alternatives you may wish to choose.

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 load the floppy into ram as '/'.  This is the
   first step in doing a normal installation or upgrade.
Type 'ro <RETURN>' to boot with the floppy as root.  This is the
   first step in installing or upgrading a machine with < 4mb of memory.
Type 'ro root=xxx <RETURN>' to mount a hard disk partition as '/'.
   The value 'xxx' is printed for you during the installation or upgrade.
   This is the second boot on machines with < 4mb of memory, and on other
   machines unless you install LILO on your hard disk.

   Typical values for root:   /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 from this floppy at least once, and possibly twice or three times:

1. You may need to run '`fdisk`' to edit the partition tables. After doing this, if '`fdisk`' reports that the kernel has not re-read the partition table successfully, you should reboot.
2. You need to set up and mount the hard disk partitions and install basic files from the boot and root disks. If you do not use the '`ramdisk`' option, you will need to reboot after installing packages from the root disk.
3. If you use the '`ramdisk`' option, you must reboot before writing a boot floppy or before installing the MCC packages from floppies in drive 0.

One important decision that you must make is this: will you use the '`ro`' option at your first boot, or will you use '`ramdisk`'? The '`ramdisk`' option has some advantages and some disadvantages:

- The ramdisk takes up 1.2 Mb of memory. This is impossible on machines with less than 4 Mb of memory, and it slows down the installation noticeably even on machines with as much as 8 Mb of memory.
- The '`ramdisk`' option may save you some fiddling, particulary if you are willing to use LILO to install the boot code directly onto your hard disk.
- The '`ramdisk`' option does not allow you to create a test boot floppy without an additional reboot.
- The '`ramdisk`' option does not allow you to install MCC packages from drive 0 until after you reboot without a ramdisk.

Considering these reasons, I prefer to think of the simpler '`ro`' option as the 'normal' way of installing MCC Interim Linux, and of the '`ramdisk`' option as the exception. The following discussion therefore treats the '`ramdisk`' procedure only as an alternative.

To boot from the boot disk without using a ramdisk, simply type '`ro`' and press `<RETURN>` after the above boot message appears. Wait until you receive this prompt:

```
VFS: Insert root floppy and press ENTER
```

Then remove the boot floppy, put the root floppy in its place, and press `<RETURN>`. You should shortly see the message '`Loading /etc/kbmap`', which is described below.

If you wish to use a ramdisk, boot from the boot disk and press `<RETURN>` after the LILO boot message. Eventually you will see these messages:

```
RAMDISK: Loading 1200 blocks into RAM disk.............................

Now remove the boot disk from the floppy drive, place the
root disk in the drive, and press <RETURN>.
```

After this the procedure is the same as for booting with the '`ro`' option, at least until partway through the installation process.

Whether you use the ramdisk or not, at some point you will wish to boot with your installed hard disk partition as root. If you have not installed boot code on your hard disk, and if you have not made a special boot floppy using LILO or some other method, you can still do this using the MCC boot disk. Indeed, if you have not used the '`ramdisk`' option, you must reboot in this fashion at least once. To do so, boot from the floppy. At the LILO prompt, you need to give a command such as this:

```
ro root=302
```

where the number '`302`' varies depending on which root partition you intend to use. If you follow the 'normal' MCC installation procedure, this command is printed for you just before the first reboot when you need to use it. Rebooting in this way uses the kernel on the boot disk, but mounts one of the partitions on your hard disk as `/`, and it mounts that partition in read-only mode. This is now the normal way to boot, since it allows the system to check and repair the root file system before any programs try to write to it.

When the kernel boots, you should see a message like this:

```
Linux version 1.0.4 (root@linux) #1 Wed Mar 30 23:49:33 GMT 1994
Partition check:
  hda: hda1 hda2 hda3 hda4 < hda5 hda6 hda7 hda8 >
```

The partition check should tell you what device you need to use when you run '`fdisk`'; for example, `/dev/hda` in the above case. Other possibilities are `/dev/sda`, `/dev/hdb`, `/dev/sdb`, `/dev/xda`, and so forth.

Whenever 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 a second hard disk; of course, this is because it does not exist. You may see similar messages which report that you do not have SCSI devices, or that certain Ethernet hardware cannot be found. You can give the kernel certain information which avoids the need to probe for equipment; for more information, see Section 5.5 [Passing parameters to the kernel], page 45.

At some point in your initial boot from the MCC boot disk, you should see the following messages:

```
Loading /etc/kbmap

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.
r   Run a customise script.
q   Quit.

? _
```

The message '`Loading /etc/kbmap`' means that the kernel is loading the default (UK) keyboard configuration file from the root disk. Then instead of getting a shell prompt '`#`', you see a small menu. The choices in the menu help you execute several commands. These commands will prepare your machine to hold Linux, and will begin the installation process when you are ready to do so. The 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 most other distributions of Linux.

Note: It is not possible to get a shell prompt using the MCC boot and root disks until *after* you have installed Linux on your hard disk. The MCC '`init`' program can then execute the `/bin/sh` on your hard disk.

4.3. Editing the partition tables

To run '`fdisk`' from the MCC root floppy, select '`f`' from the menu. See Section 6.2 ['`fdisk`': the Linux partition table editor], page 50, 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 before quitting.

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 of the output printed on 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   83  Linux native
/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): _
```

The partition type (by default 83 for Linux native file systems) is actually ignored in practice (unless you are using DR-DOS), but I like to see things clearly when I look at partition tables; I want sensible information to appear. If I have a Linux/MINIX file system, I can change the partition type as follows:

```
Command (m for help): t
Partition number (1-8): 3
Hex code (type L to list codes): 81
Changed system type of partition 2 to 81 (Linux/MINIX)
```

Next we make the new partition active. In most cases this has no practical effect, at least if you install LILO in your master boot record. If you use the DOS master boot record, you must have exactly one active partition, while other programs (such as the OS/2 boot manager) have their own rules. See Section 6.2.5 [Active flags and system types], page 57. 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 native
/dev/hda3   *     263     263     426   40180   83  Linux native
/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): v
238664 unallocated sectors.
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
  hda: hda1 hda2 hda3 hda4 < hda5 hda6 hda7 hda8 >
Syncing disks.
```

Remember that it is a good idea to use the '`verify`' command before saving the tables. The '`unallocated sectors`' message is just a warning; after all, we have no partitions in the area from cylinder 492 to cylinder 977.

If everything works properly, you should see a list of available partitions after the line '`Calling ioctl()`' and before the line '`Syncing disks`', just as in the above example. But sometimes you will see a message like this one:

```
Re-read table failed with error 16: Device or resource busy.
Reboot your system to ensure the partition table is updated.
```

This means that, if you have changed the partition tables, the kernel has not read them. In this case you may run '`fdisk`' again without rebooting the system, but you *must* reboot before running '`mkswap`', '`mkfs`', or installing Linux. To reboot at this stage, give the answer '`q`' at the menu prompt.

If the warning message does not appear, it is perfectly safe to continue the installation without rebooting.

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, so that you don't run out of memory. 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 root 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. If you need more than 16 Mb of swap space, you may have up to 16 swap partitions and swap files, each up to 16 Mb in size.

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 may make the installation a little faster. Once Linux is installed, you can activate swap space with the '`swapon`' command, or put an entry in `/etc/fstab` to activate it automatically at boot time. To run '`swapon`' from the MCC root 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 root 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 an ext2 file system (Y or N): y
mke2fs 0.5 ALPHA, 15-Mar-94 for EXT2 FS 0.5, 94/03/10
```

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. A *minix* file system cannot have more than 64 Mb, or 65536 blocks of 1024 bytes, on a single partition.

Note: As of version 1.0+, MCC Interim Linux will not install properly on a *minix* file system unless the maximum length of filenames is 30 characters. If you have an older *minix* file system, back it up and run '`mkfs`' before installing. In any case, we do not recommend using the 14 character version unless you need to maintain compatibility with the MINIX operating system.

The *ext2* file system was developed for Linux and will eventually become the normal 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 an *ext2* file system, it assumes that it is a *minix* file system with 30 character filenames.

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 root floppy, select '`c`' from the menu. Here is a typical example:

```
Run fsck on (block device; default = /dev/hda1): /dev/hda3
Does this device contain an ext2 file system (Y or N): y
Check for bad blocks (Y or N): y
Do you want automatic fixing (Y or N): y
Running command: badblocks -s /dev/hda3 40180
Peak memory: Memory used: 143360, elapsed time:  9.341/ 0.660/ 2.020
```

The check for bad blocks is not strictly necessary, but it is a good idea to check older disks once in a while. This check is not available for *minix* file systems.

Just as when you create a file system, the root 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 two 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.

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 '`n`' to the query about automatic fixing, '`fsck`' runs in interactive mode; otherwise, it runs in automatic 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 when you reboot, you will lose all or part of the file irrevocably.

4.5. Installing basic files from the MCC disks

The description of a basic installation or upgrade can be simplified if we consider the read-only and ramdisk procedures separately.

4.5.1. Installing base packages using the '`ro`' option

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 an ext2 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:

```
654 blocks
Installing Base.tgz
Base (part 1) contains basic binaries and text files for Linux.
.....................................................................
2701 blocks
Making holes in lib/ld.so...
There are 4096 byte holes out of 17412 bytes in 'lib/ld.so'.
```

```
Making holes in lib/libc.so.4.5.21...
There are 349184 byte holes out of 623620 bytes in 'lib/libc.so.4.5.21'.
Making holes in libm.so.4.5.21...
There are 94208 byte holes out of 107524 bytes in 'libm.so.4.5.21'.
HOSTNAME: "linux"
Verifying Base.crc.
```

The 'blocks' messages refer to 512-byte blocks, half the size of the file system blocks. The '`Making holes`' messages come from the '`makehole`' program, which allows us to save nearly half a megabyte of disk space by converting the shared libraries to 'sparse files'.

'`Verifying`' refers to the cyclical redundancy check by the '`brik`' program. 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 sbin/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) tells you how full your file systems are getting. This can be useful if you have insufficient space to install further packages.

Usually you will reboot the system at this point during an installation or upgrade. If you choose not to do so, you can start a shell, and you can explore the half-installed system. If you do this, 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:

```
Unmounted /dev/hda3
Unmounted /dev/hda2

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

If you have just installed LILO on your hard disk, then remove any disk
from the floppy drive.  Otherwise, place the boot floppy in the drive.
Then press <RETURN>.  If the system doesn't reboot, press the RESET button.
```

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 before pressing `<RETURN>`. The most 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.

Sometimes the reboot at this point fails, especially if you remove the floppy disk from the drive as instructed. I have left the instructions as they are, simply because most people would find it a bit tricky during their first installation to fumble the root disk out of the drive, and the boot disk into the drive. If the reboot fails on its own, simply press the `RESET` button, or power off and then power on again.

After you reboot and type the '`rw root=`*xxx*' command at the LILO prompt, leave the boot disk in the floppy drive so that installation can continue. You should see the following messages:

```
140 blocks
Installing base.tgz
Base (part 2) contains basic text and binary files.
.............................................................
2665 blocks
Verifying base.crc.
```

At this time the kernel and its `psdatabase` are installed, followed by part 2 of the base package. Then you are asked this question:

```
Do you wish to change the default keyboard map (Y or N): _
```

The default keyboard map is the one on the root disk, which is for the UK keyboard unless you have changed it. If you have access to a Linux system, you can take any keyboard map acceptable to the kbd-0.85 utilities and copy it onto the root disk before installation, replacing the etc/kbmap file which is already there. This will make your national keyboard available during all stages of the installation process. I expect various ftp sites in countries with keyboards quite different from the UK keyboard will be willing to make a local version of the root disk available.

If you do not wish to keep the default keyboard map, or if you are uncertain, answer the question with '`y`', and you will see this message:

```
 1  be-latin1     8  es          15  gr-latin1    22  sf
 2  defkeymap     9  es2         16  no           23  sf-latin1
 3  dk           10  fi          17  no-latin1    24  sg
 4  dk-latin1    11  fi-latin1   18  no-latin1+   25  sg-latin1
 5  dvorak       12  fr          19  pl           26  uk
 6  dvuk         13  fr-latin1   20  ru           27  us
 7  emacs        14  gr          21  russian

Select the default keytable for this machine (0 to quit): 26
Loading etc/kbmap
```

These list the keyboard map files installed in part 2 of the base package. The file corresponding to the number you select is copied to `/etc/kbmap`, where it is loaded each time your system boots.

Next you are able to install the *tcpip* package:

```
Do you wish to install networking software (Y or N): y
Installing tcpip.tgz
Tcpip contains networking programs, daemons, and configuration files.
These will not work unless your kernel was compiled with tcpip.
.....................................................................
1470 blocks
Compressing formatted man pages
Verifying tcpip.crc.
```

The networking software should work even if you do not have Ethernet hardware, as long as networking is configured into your kernel. In any case you must configure the networking software before it can run. You may do this at any time by giving the '`install.net`' command, but the installation procedure will configure it now if you wish.

```
You may configure networking even if you have no Ethernet card
by selecting the default IP address 127.0.0.1.  If you have a
supported Ethernet card, you need to know your real IP address,
and you should know the address of your gateway machine and of
a domain name server before you configure it.  If there is no
gateway or name server machine, you may use 127.0.0.1 for it.

Configure networking now (Y or N): _
```

If you answer '`y`', then the software is configured exactly as described in Section 5.3 [Configuring the *tcpip* package], page 42. If you answer '`n`', the following message appears:

```
The 'install.net' command will configure networking for you.
```

The installation continues by giving you the opportunity to install LILO boot code either on a floppy or on your hard disk. See Section 4.7.1 [Installing LILO on a floppy or hard disk], page 35. If you do not wish to do this, you may create a conventional boot floppy instead. See Section 4.7.2 [Making a conventional boot floppy], page 37.

If you do not create a boot floppy or install LILO boot code 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. You may, of course, continue to use the MCC boot disk with the command '`rw root=`*xxx*' at the LILO prompt.

At this point in the installation you may install other MCC packages if you wish. See Section 4.6 [Installing the MCC Interim packages], page 32.

4.5.2. Installing base packages using the '`ramdisk`' option

If you have used the '`ramdisk`' option at boot time, you must take the same initial steps that you take when you use the '`ro`' option: set up partitions, make swap space, make file systems, and select the menu option to install with the '`i`' command. This section explains the differences between the subsequent stages of an installation or upgrade using the '`ramdisk`' option, and the subsequent stages using the '`ro`' option, as explained above. See Section 4.5.1 [Installing base packages using the '`ro`' option], page 28.

After part 1 of the base package has been installed, part 2 is installed automatically, without rebooting. You may install and configure the *tcpip* package as well. If you install the networking software without configuring it at this time, you should not give the command '`install.net`' until after you have rebooted, and a message warns you of this during the installation process.

After you have installed or omitted the *tcpip* package, you may wish to configure LILO. You should not remove the root disk from drive 0; therefore you cannot store the LILO boot code on a floppy at this time. If you wish to make a floppy to test the LILO installation before writing the boot code to your hard disk, reboot first, log in as root, and give the command '`install.lilo`'. If you wish to install the LILO boot code on a test floppy, you may do so now. If you are confident that it will work, you may install it directly on your hard disk, without creating a test floppy.

At the end of the basic installation, you are allowed to install MCC Interim packages, just as at the end of the installation which follows the '`ro`' procedure. Because you are not allowed to remove the root floppy from drive 0 at this time, you may not use that drive to install the packages. If you wish to install packages from floppy disks in drive 0, or from a directory which you will mount using NFS, reboot the system, log in as root, perhaps mount a remote file system using NFS, and give the command '`/tmp/bootinstall`'. You may install packages at this point from a floppy in drive 1, from a directory already mounted during the '`install`' procedure, or from a directory in an unmounted MS-DOS, *minix*, or *ext2* file system on the same machine.

If you do not wish to reboot immediately after installation, you may start a shell on the newly installed partition and browse. You should bear in mind that many commands will not work properly, since the root disk is not mounted as `/`, but as `/root`.

When you have browsed, leave the shell with the '`CTRL-D`' or '`exit`' command, and answer '`y`' to the question about rebooting. You will see a message

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

in case you have not installed the LILO boot code on your hard disk. As with the '`ro`' procedure, the reboot may fail at this point. If it does, press the `RESET` button or, if necessary, switch the power off and on again.

4.6. Installing the MCC Interim packages

After you have installed the packages on the boot and root disks, you probably wish to install other MCC Interim packages. You can do this in one of three ways:

- The MCC installation allows you to install packages immediately after installing the basic packages. Certain restrictions apply at this point if you chose the '`ramdisk`' option at boot time. See Section 4.5.2 [Installing base packages using the '`ramdisk`' option], page 31.
- The MCC distribution installs the command '`/tmp/bootinstall`', which allows you to install many packages from floppies or from a single directory. This is most useful for installations using NFS, or when you wish to test installed LILO boot code before installing packages.

– The MCC distribution installs the command '`mccinstall`', which allows you to install one package at a time.

In the present section we discuss the first and second of these ways of installing packages. Whether you give the '`/tmp/bootinstall`' command explicitly, or have the installation process give it, you should see the following message:

```
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 in the middle of a '`ramdisk`' installation, options 1 and 2 do not appear.) 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 that directory is currently mounted, 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 have networking software installed, configured, and running before you can mount the remote file system. Normally the networking software is not running during the MCC installation, and you should not start it if you have a ramdisk mounted as `/`. If you did not configure networking software during the basic MCC installation, give the command '`install.net`' and run '`/etc/rc.net`' before trying to mount a file system from another machine.

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 chose previously:

```
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 are installed from the specified directory or floppy drive. The messages which appear are similar to those for the basic packages on the root and boot floppies, and include the following information:

```
Baseman contains formatted man pages for the base package.
Bison contains the GNU yacc-compatible parser generator.
Elisp contains lisp files for GNU emacs 19.22.
Emacs contains GNU emacs 19.22, except for lisp files.
Emacsxtr contains the portions of GNU emacs which are not contained
    in other emacs packages in the MCC Interim distribution.
Extrainf contains extra info files for bison, emacs, gas, gcc, texinfo,
    and gdb.
Extralib contains additional libraries for gcc, release 2.5.8.
Flex contains the GNU clone of lex, the lexical analyser generator.
Gawk contains the GNU version of the awk programming language.
Gcca contains executables, include files, and man pages for gcc 2.5.8.
Gccb contains libraries and internal binaries for gcc, release 2.5.8.
Gdb contains the GNU debugger.
Gpp contains the GNU C++ compiler, release 2.5.8, except for those
    files contained in the gcc packages.
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 1.0.
Lp contains programs and manpages for printing.
Mail contains binaries and support files for smail and for elm.
Manpages contains all unformatted manual pages which are distributed
    (formatted) as part of MCC Interim linux.
Patches contains all patches used to compile the binaries included in the
    current MCC Interim distribution.
Progman contains formatted manual pages from sections 2, 3, and 9.
Timezone contains data for world timezones and related programs.
Words contains a /usr/dict/words file.
The kernel in ipide has no SCSI or xd support.
The kernel in wd has no SCSI or xd support.
    It contains support only for Western Digital 80*3 Ethernet cards.
```

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 allow these to appear, since they might sometimes contain useful information.

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 to boot other operating systems. 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 '`lilo`' command, with its support files, is contained in part 2 of the MCC *base* package. The support files include `/etc/README.lilo`, which describes many of the features of this utility, and `/etc/lilo.conf`, the configuration file which you may wish to edit. See Section 5.4 [Editing the system configuration files], page 44. The binary support files for LILO are now installed in the `/boot` directory. The MCC installation procedure runs the command '`install.lilo`', a simple utility for installing LILO boot code, which asks you the following questions:

```
The LILO utility allows you to boot Linux from a hard disk.
It can give you the choice of booting one or more versions of Linux,
or of booting one or more other operating systems.  LILO installs a
boot program in any partition (a complex option, since you must set
active flags properly), or in the Master Boot Record (a simple option),
or on a floppy disk (a safe option, since you can test for problems).
Would you like LILO to install a boot program now? y
```

```
Where should the boot code go?  You may override this later to put the
boot code on a floppy instead of the hard disk.  Possible choices
include /dev/hda (the master boot record on your first hard disk),
/dev/hda2 (the partition you are installing Linux on),
or /dev/fd0 (floppy disk 0, DOS floppy A:).
Give me the name of a block device, or 'q' to quit.
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.

At this point, LILO needs to construct a 'database' of bootable images and systems. This information is saved in the file `/etc/lilo.conf`, which tells the LILO installer what to install, where, and how. Initially this file contains no systems or images; you must add each one which you wish LILO to be able to boot, including the system you have just installed.

```
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 'L' for Linux, 'o' for other systems, 'q' to quit:  (L, o, q): l

The default Linux image in this installation is /boot/vmlinuz.
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 'L' for Linux, 'o' for other systems, 'i' to install:  (L, o, i): o

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 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 'L' for Linux, 'o' for other systems, 'i' to install:  (L, o, i): i
```

This little loop repeats, expecting the answer '`L`' for a Linux image, '`o`' for another operating system, and '`i`' when you have no more options to add. You must be careful not to give the same name to two options. The '`L`' can be typed in lower case, but the prompt uses upper case because '`l`' looks too much like the digit '`1`'. The file `/boot/vmlinuz` is installed from 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 `/boot/vmlinuz.old` if it exists when LILO is run. The MCC extra 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.  You may install it in /dev/hda.
Give me the name of a block device, or 'q' to quit.
The default is /dev/fd0: /dev/hda
Added linux
Added dos
Skipping /boot/vmlinuz.old
```

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/disktab` to insert the correct geometry for your disk, then give the command '`/etc/lilo/newinstall`' to try again. The '`lilo`' command usually asks you to edit this file if you need to do so. The file itself contains a sample entry, and the file `/etc/README.lilo` contains more information about the `/etc/disktab` file.

The boot floppy produced at this point will allow you to test whether the LILO boot code works properly before you attempt to install it on your hard disk. After rebooting from the test floppy, log in as root and give the command '`lilo`' to place the boot code in its permanent home.

Sometimes it is necessary to pass disk or other parameters to the kernel explicitly. These can be included in the file `/etc/lilo.conf` so that you do not need to type them explicitly every time you boot Linux. See Section 5.5 [Passing parameters to the kernel], page 45.

If you do install LILO boot code 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 `/boot/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; in other words, this command can delete partitions or make them inaccessible if you have changed your partition table since it was first created.

If you have MS-DOS version 3.3 or later, you can create a normal DOS Master Boot Record with the DOS command '`FDISK /MBR`'. Be certain that the DOS partition is marked active (by using '`fdisk`') before you give this command.

4.7.2. Making a conventional boot floppy

If you do not install LILO boot code on your hard disk during the MCC installation, you may wish to make a conventional boot floppy. The '`ro`' installation procedure asks you this question if you do not choose to install LILO boot code:

```
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=/boot/vmlinuz of=/dev/fd0 bs=3072
```

The resulting disk should boot under any circumstances. Unfortunately, you cannot pass parameters to this kernel, as you can if you use LILO to boot; thus this disk is not useful if you have a difficult hard disk or some other hardware problem. But for most of those people who cannot get LILO to work, or who prefer not to use it, this is a way of booting Linux without messing about with the partition tables. This should not really be necessary, since there are very few known problems with LILO. Of course, you may need to put correct parameters in `/etc/disktab` for a few awkward disks, or to use the 'append' feature to pass parameters to the kernel. See Section 5.5 [Passing parameters to the kernel], page 45.

4.7.3. Making a special boot floppy using LILO

It is useful to be able to create a floppy which boots, and which can be used as a recovery floppy or to boot with your hard disk as root. It is hard to fit all the files you may wish onto a floppy, particularly if you must boot from a 5.25 inch floppy. The following instructions should work in the simplest cases.

First, create a file, for example `/tmp/lilo.conf` containing the following lines:

```
install = /mnt/boot.b
delay = 20
compact
vga = normal
backup = /dev/null
map = /mnt/map
boot = /dev/fd0
image = /mnt/vmlinuz
  root = /dev/fd0
  ramdisk = 0
  read-write
```

Then place a new floppy in drive 0, and give the following commands:

```
# fdformat /dev/fd0H1440             # for a 3.5 inch floppy
# mkfs.minix -c -n14 /dev/fd0 1440
# fdformat /dev/fd0h1200             # for a 5.25 inch floppy
# mkfs.minix -c -n14 /dev/fd0 1200
# mount -t minix /dev/fd0 /mnt
# mkdir -p /mnt/etc /mnt/lib /mnt/bin /mnt/dev /mnt/mnt
# ln -s bin /mnt/sbin
# cp -p /boot/boot.b /boot/vmlinuz /mnt
# rdev /mnt/vmlinuz /dev/fd0
# rdev -R /mnt/vmlinuz 0
# cp -p /lib/ld.so /lib/libc.so.4.5.21 /mnt/lib
# mv /mnt/lib/libc.so.4.5.21 /mnt/lib/libc.so.4
# (cd /bin;tar cf - cat chown cp dd df ln ls mkdir mount mv \
 rm rmdir sh sync umount)|(cd /mnt/bin;tar xfp -)
```

```
# (cd /dev;tar cf - console fd0 fd1 null tty tty? hd* sd* xd*)|\
 (cd /mnt/dev;tar xfp -)
# (cd /sbin;tar cf - badblocks fdisk fsck.* mkfs.* mkswap swap* \
 tune2fs)|(cd /mnt/bin;tar xfp -)
# echo "/dev/fd0 / minix rw 0 0" >/mnt/etc/mtab
# lilo -C /tmp/lilo.conf
Added vmlinuz
# df /dev/fd0
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/fd0                1183    1078      105     91%   /mnt
# umount /mnt
```

This example used a small kernel (`vmlinuz`) 252420 bytes in size. If your kernel is much larger (for example, the 363012 bytes of the MCC kernel with CD ROM drivers included), it will be too big for a 5.25 inch floppy with all of the above commands.

If you create and boot from this floppy, you must be careful not to leave the file system on it dirty. When you are finished using it, give the command '`umount -a`' before pressing `CTRL-ALT-DEL` or the `RESET` button.

5. 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 this chapter.

5.1. 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 add drivers or file systems, to add some of the available patches, to remove features you may not need, such as maths coprocessor emulation, or to add support for more than 16 Mb of RAM.

To recompile the kernel, you should install the *gcca*, *gccb*, and *linux* packages from the MCC distribution. Then '`cd`' to `/usr/src/linux-mcc`. Then type '`make config`' and answer the questions with '`y`' or '`n`'. Finally give these commands:

```
make dep
make zlilo
```

The last command installs the newly compiled kernel in /boot and uses LILO to prepare to boot from it. It also runs the command '`psupdate`' to change the file `/etc/psdatabase` so that the '`ps`' commands work with the new kernel. The '`make zlilo`' command also saves the old `/boot/vmlinuz` as `/boot/vmlinuz.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 that the commands for the '`zlilo`' target have been altered slightly in the MCC Makefile to conform to the conventions of this distribution: the variable '`VMLINUZ`' has been added, and the copying of `zSystem.map` removed. You may wish to make similar changes to the Makefile which comes with later versions of the Linux kernel:

```
VMLINUZ = /boot/vmlinuz
zlilo: $(CONFIGURE) zImage
     if [ -f $(VMLINUZ) ]; then mv $(VMLINUZ) $(VMLINUZ).old; fi
     cat zImage > $(VMLINUZ)
     /usr/bin/rdev -R /boot/vmlinuz 1
     /usr/sbin/psupdate
     if [ -x /sbin/lilo ]; then /sbin/lilo; else /etc/lilo/install; fi
```

Note: This release installs 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 the C compiler works without installing the kernel sources, and 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/include
(cd /usr/include;tar cf - asm linux)|(cd /usr/src/linux/include; \
 tar xvfp -)
rm -rf /usr/include/asm /usr/include/linux
ln -s /usr/src/linux/include/* /usr/include
```

Note that this is not necessary; the kernel source installed from the *linux* package should compile without changing anything.

5.2. Setting up Ethernet cards

The 'standard' kernel as distributed by Linus Torvalds currently contains support for a limited number of Ethernet cards. See Section 1.1 [What is Linux?], page 1. Western Digital cards 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 use it to configure the card in case '`wdsetup`' doesn't work. (The '`EZSETUP`' command is supplied with most Western Digital cards on a DOS floppy. If you don't have such a floppy, ask the dealer from whom you bought the card.) The manual page for '`wdsetup`' is installed as part of the *tcpip* package, and contains more information about this program.

Login as root, then '`cd /etc`'. 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 -i 5
```

to set the card's address and IRQ to appropriate values. The current kernel source sets the IRQ for some tape devices to 5, which conflicts with the most usual Ethernet card IRQ. Either change the kernel source, or change the Ethernet card's IRQ to another value, e.g., 10. If this doesn't work, you need to read the file `README.wdsetup` for more information.

Besides changing the cards themselves, the new net-2 software allows you to configure the drivers in the kernel to work with your card. You need to do this only if the kernel fails to recognise your Ethernet card at boot time, and writes the message '`No ethernet device found`' on your screen and in the file `/var/adm/messages`. In this case, after running '`install.net`', read the manpages for the commands '`ifsetup`' and '`iflink`'. As I understand it, you will need to use '`ifsetup`' to reconfigure the driver, and then user '`iflink`' to link the driver to `/dev/inet`. Unfortunately, the net-2 FAQ file contains no information about these commands, and I do not find the man pages clear. Once you determine the correct form of these commands for your machine, edit `/etc/rc.net` and insert them where the note about '`ifsetup`' appears.

5.3. 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. All kernels distributed in this MCC release include both TCP/IP and NFS. This software should work even if you

have no Ethernet card. In that case, of course, you cannot contact other machines, unless you use SLIP or PLIP. This MCC release does not contain the '`dip`' software usually used to start SLIP connections. More information about SLIP appears in the Linux networking HOWTO. See Section 7.3 [The Linux HOWTO documents], page 66.

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 many such patches is available by anonymous ftp from `ftp.mcc.ac.uk` in the directory `/pub/linux/newether`.

Before you configure the *tcpip* package, you should know your hostname, domain name, IP address, netmask, gateway address, and the address of a Domain Name Server. In particular, you may need to have your network manager assign you an IP address. If you have no Domain Name Server, or no gateway, you may use the loopback address `127.0.0.1` for these values.

After installing the *tcpip* package, give the command '`install.net`'. (This is done during installation if you answer '`y`' to the question about configuring networking software.) The following dialogue ensues:

```
# install.net
What is your hostname (default = linux): avl0
Internet address of avl0 (default = 127.0.0.1): 130.88.201.62
Domain name for avl0 (default = (none)): mcc.ac.uk
Netmask for avl0 (default = 255.255.255.0): _
Gateway address (default = 130.88.201.250): _
IP address of your domain name server (default = 130.88.13.7): _
Hostname:         avl0
Domain name:      mcc.ac.uk
Fully qualified:  avl0.mcc.ac.uk
Internet address: 130.88.201.62
Netmask:          255.255.255.0
Network address:  130.88.201.0
Gateway address:  130.88.201.250
DNS server:       130.88.13.7

Is this correct (Y or N): y
HOSTNAME: "avl0"
You may add more nameservers by editing /etc/resolv.conf
Networking software is now configured.
```

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`' or '`ifsetup`'. See Section 5.2 [Setting up Ethernet cards], page 42. Note that the '`install.net`' script is aware of the peculiar gateways and netmasks used in Manchester, and makes what ought to be sensible guesses at default values for other sites.

After installing networking software, you should edit the file `/etc/inetd.conf` to comment out any services you wish not to allow. To remove a service, insert an initial '`#`'. If '`inetd`' is already running, you must send it SIGHUP or reboot after changing this file; otherwise the changes have

no effect. Some 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`.

Other networking programs are available from the various Linux ftp sites. They should compile and fit into the MCC installation with little difficulty. Note that MCC does not use shadow passwords, so any programs which have the option of shadow password support must be compiled without it.

Note: These programs are compiled from the most recent source I could find. I have fixed a few bugs, but there may still be more. The source I actually compiled can be found (with no patches needed) in the file `netsrc25m` in the directory `sources` in the current MCC Interim distribution.

5.4. 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 is intended to contain local commands which you want to be run at boot time. The distributed file contains the '`selection`' command (commented out). This command will not work unless you have an appropriate `/dev/mouse` device. For a serial mouse, this is usually the same as `cua0` or `cua1`. For a bus mouse, this is usually the same as one of the devices `logibm`, `psaux`, `inportbm`, `atibm`. Link the appropriate device to `/dev/mouse`, for example with the command '`ln /dev/cua0 /dev/mouse`'. Note that no MCC kernel comes with support for bus mice already compiled, and that some bus mice will cause problems if '`selection`' is running at the same time as the X windowing software.

`/etc/mtools`
: See Section 6.4 [Special commands for DOS floppies and partitions], page 60.

`/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.conf`
: This contains the command which controls what happens whenever LILO is reinstalled. You must give the command '`lilo`' whenever the system file (`/boot/vmlinuz` by default) is modified (for example, by the '`rdev`' command) or moved: LILO does not use the directory structure to find the file. The file `/etc/README.lilo` contains information about the contents of the file `/etc/lilo.conf`.

`/etc/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`
: Some services for inetd are commented out. See Section 5.3 [Configuring the *tcpip* package], page 42.

`/etc/kbmap`
: This file contains the key definitions for your keyboard. By default the British (UK) keyboard map is installed. Other available keymaps are installed in part two of the *base* package, and the MCC installation gives you the opportunity to change your default. The command '`kbdinstall`' allows the root user to change the system default keyboard map.

  Some users with special requirements have asked whether the Shift, Control, and Alt keys can be changed to toggles. To do this, simply edit the file '`/etc/kbmap`', changing the keysyms '`Shift`', '`Control`', and '`Alt`' to '`Shift_Lock`', '`Control_Lock`', and '`Alt_Lock`' respectively. If you do this, it is a good idea to set some key (for example, Caps_Lock or a function key) to Alt, to avoid problems which occur when switching between virtual consoles.

5.5. Passing parameters to the kernel

It is possible to pass parameters to the kernel in two ways: by specifying them at the LILO prompt, or by an option in the file '`/etc/lilo.conf`' such as '`append = "hd=202,64,32"`'. These options may be useful if your kernel does not detect your hardware configuration completely, or in other circumstances. Unfortunately they are not widely or clearly documented.

Among the parameters accepted by MCC kernels are these:

`aha152x=portbase,irq,scsiid,reconnect`
: Four integers required for Adaptec AHA-152x driver.

`bmouse=mouse_irq`
: One number describing a bus mouse.

`debug` Set the console log level to 10.

`ether=irq,base_addr,mem_start,mem_end,name`
: One to four numbers and a device name for Ethercard drivers. All arguments are optional, with *name* being the first non-numeric argument.

  irq `0` means to use autmatic detection.

  base_addr `0` means to probe.

  mem_start Some drivers use this to specify the debug level: `0`–`7`, with 0 meaning no messages, and 7 the maximum.

  mem_end The 3c502 driver uses this to select between internal and external transceivers: `0` for internal (the default), `1` for external. The *name* argument should be, for bus-attached Ethernet cards, one of the following:

`eth0`, `eth1`, `eth2`, `eth3`. The following names are accepted for parallel port adaptors: `atp0`, `dl0`.

`hd=cyl,head,sector`
: Three numbers describing geometry of the (first) hard disk.

`mcd=port`
`mcd=port,irq`
: One or two numbers describing Mitsumi CD ROM.

`ncr5380=port,irq,dma`
: Three numbers required for NCR 5380 driver. Because of a bug, it appears that only the first two may be specified.

`no387`
: There is no maths coprocessor on this machine.

`pas16=io_port,irq`
: Two numbers required for Pro Audio Spectrum/Studio 16 driver. The kernel doesn't actually allow this to be specified at the moment.

`ramdisk=`*nnn*
: Reserve *nnn* Kb of memory for a ramdisk. The MCC kernels have the following patch applied to `linux/init/main.c` to enable this option:

```
@@ -298,4 +298,6 @@
                } else if (!strcmp(line,"ro"))
                        root_mountflags |= MS_RDONLY;
+               else if (!strncmp(line,"ramdisk=",8))
+                       ramdisk_size = simple_strtoul(line+8,NULL,0);
                else if (!strcmp(line,"rw"))
                        root_mountflags |= ~MS_RDONLY;
```

`reserve=start,extent{,start,extent}`
: One or more pairs of numbers specifying the base address of a reserved region of memory and the size of that region. This prevents other drivers from probing the region unless another kernel argument explicitly tells them to do so. For example, '`reserve=0x300,32, ether=0,0x300,eth0`' keeps all device drivers except the Ethernet card drivers from probing `0x300`–`0x31f`.

`ro`
: The root file system is to be mounted in read-only mode.

`root=`*nnn*
`root=/dev/`*xxx*
: Specify the hex number or device name for the root file system partition.

`rw`
: The root file system is to be mounted in read-write mode.

`sbpcd=ioaddr,type`
: One number and a string for the IDE-style Soundblaster/Panasonic CD ROM driver. The *ioaddr* might be, for example, `0x230`. The string is case sensitive, and may be either '`LaserMate`' (the default) or '`SoundBlaster`'.

`sound=`*integers*
: It is not clear to me from `drivers/sound/dev_table.c` exactly how many integers are passed, or what they mean.

`st0x=base_address,irq`
: Two numbers required for Seagate ST01/ST02 SCSI driver.

`t128=address,irq`
: Two numbers required for Trantor T128/T128F/T228 driver.

`tmc8xx=base_address,irq`
: Two numbers required for Future Domain TMC-885/TMC-950 SCSI driver.

`xd=type,irq,iobase,dma`
: Four numbers required for XT hard disk controller (`/dev/xda`).

5.6. Installing the XFree86 software

This release of Linux is compatible with XFree86 release 2.1. The software is available from ftp sites; for example, from `ftp.mcc.ac.uk` in the directory `/pub/linux/packages/X11/XFree86-2.1`. This directory contains two files, `INSTALL` and `README.Linux`, which you should read with care.

Select the files you need: one of the seven server files, together with the six other required files, form a minimum installation, as described in `README.Linux`. The required files are small enough to fit on floppy disks. Place these files in a directory somewhere on your Linux machine, e.g., in `/tmp`. Then give the following commands:

```
cd /usr/X386
for i in /tmp/XF86-2.1-*.tar.gz;do tar xzfp $i;done
```

Note: Be sure to include the '`p`' modifier on the '`tar`' command, or some of the binaries will not work properly.

After the software is installed, you must give the command '`ldconfig`', so that programs can access the new shared libraries. Only then should you tackle the job of creating an `Xconfig` file for your machine.

Producing a suitable `Xconfig` file tricky, since it is dependent on both your graphics chip and your monitor specifications. The instructions in the file `/usr/lib/X11/etc/README.Config` should be followed *very carefully*, since you can damage your monitor if you specify values which are very far from the correct ones for your hardware. Calculating the correct values for the `Xconfig` file is the most difficult aspect of installing X windowing software.

Once the XFree86 software is installed and configured, you will probably want to fine-tune the installation. If you are using the SVGA server, the command '`vgaset`' may be useful for this purpose. It is not part of the standard X distribution, but the source is available from most ftp sites. A binary for this command can be found at `ftp.mcc.ac.uk` in `/pub/linux/binaries-4.5.21/usr/sbin`.

The current version of XFree86 sets up the same keyboard mapping which is current when X is started, so that it is no longer necessary to have a special `Xmodmap` file if you do not have a US keyboard.

When X is properly configured and fine-tuned, you may wish to change your Linux installation so that it starts X when the system boots and uses '`xdm`' to log in from the console. To do this, edit the file `/etc/inittab` and change the '`initdefault`' line to read

```
id:5:initdefault:
```

The files for configuring '`xdm`' are contained in the `/usr/lib/X11/xdm` directory. You may wish to edit `Xsetup_0` to remove the console from the '`xdm`' login screen, or to change the default paths in `xdm-config` to conform more closely to the new file system standard. Among the entries which may be useful are these:

```
DisplayManager.errorLogFile:  /var/adm/xdm-errors
DisplayManager.pidFile:  /var/X11/xdm-pid
DisplayManager.authDir:  /var/X11
DisplayManager.*.userPath:  /usr/local/bin:/usr/bin:/usr/bin/X11:/bin:.
```

You may also find it useful to set '`DisplayManager.*.systemPath`', the default `PATH` variable for root, to include `/sbin:/usr/sbin`. Some examples of files configuring `xdm` can be found at `ftp.mcc.ac.uk` in the `/pub/linux/binaries-4.5.21/usr/X386/lib/X11/xdm` directory.

6. Using Linux

6.1. Caring for your Linux system

Linux is an operating system which must be handled more carefully than DOS, or at least more carefully than 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 two methods you might use:

1. Reboot the system either by pressing '`CTRL-ALT-DEL`', or by giving the command '`shutdown -r` *time*'.
2. Stop the system by giving the command '`shutdown -h` *time*'.

The '`shutdown`' command can only be given by the user 'root'. The format of the *time* argument is discussed in the manual page for '`shutdown`'; '`now`' is a common example.

Floppy disks may be corrupted if you remove them from the drive before unmounting them, since they too might not be up-to-date at the time. Should this happen, you can repair them with the command '`fsck`', unless they contain an *msdos* file system. MCC Interim distributions are configured to run '`fsck`' on all mounted file systems at boot time. Since both *minix* and *ext2* file systems now have 'clean flags' set when they are cleanly unmounted, the '`shutdown`' command should not ordinarily be given the '`-f`' option.

There are a few files, system logs and such, which grow without limit. The logs in `/var/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 `/var/adm/wtmp` will grow constantly, and you may need to shorten it. To do this, log in as root and give the command

```
> /var/adm/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`' does 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. Some recommendations can be found in *Linux Installation and Getting Started*; for further information, see Section 7.5 [Other information about Linux], page 67.

6.2. `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.

6.2.1. 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.
3. It usually contains code which is executed when the system boots up.

The primary boot sector 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. (Note that LILO and other boot loaders can boot logical partitions, but only if the initial boot code is stored in the Master Boot Record, in one of the primary partitions, or on another disk.)

6.2.2. 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 back up 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 file 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. But even ordinary users may be allowed to mount removable hard disks and floppy disks.

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 15.

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.

6.2.3. 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. Do not use the DOS '`FDISK`' to create Linux partitions.

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 less 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   83  Linux native
/dev/hda5         837     837     936    4249+  82  Linux swap
/dev/hda6         942     942     977    1522    1  DOS 12-bit FAT
```

There are five 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 25.5 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   83  Linux native
/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`'.

6.2.4. 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. If you delete partition 2 in the above example, this deletes partitions 5 and 6 as well.
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 cylinder (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 so 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 cylinders, 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:

```
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

If there are no further messages, the kernel has successfully copied the information from the partition table into its own internal table. But sometimes you will see a message like this one:

```
Re-read table failed with error 16: Device or resource busy.
Reboot your system to ensure the partition table is updated.
```

In this case, depending on what you have changed in the partition table, it may be dangerous to continue working without rebooting, since you may lose or corrupt your data.

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. This is true even after you receive the message warning you to reboot.
2. It is not a good idea to run any of the programs '`mkfs`', '`mkswap`', '`mount`', or '`swapon`' if you have received the warning message but have not rebooted. In this case it is dangerous to run any program, but these in particular may cause serious damage to the data on your disk, including the partition tables themselves.

6.2.5. 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 is inactive, it will be flagged as active; if it is active, it will be flagged as 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 native	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 '`fdisk`' uses a type of 83. It used to use 81, the type code used by the MINIX '`fdisk`'. It seemed prudent to change the default since (a) many Linux 'minix' file systems are no longer compatible with MINIX, (b) the *ext2* file system, a native Linux file system, is fairly stable, as is the Xia file system, and (c) the number 81 causes problems with DR-DOS. Linux does not usually care what values you use for type codes, but other systems, in particular DOS, OS/2, and DR-DOS, may.

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.

6.2.6. 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. Most users do not need this data. The information 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, the *logical* or *virtual* geometry. **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.

6.2.7. 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 to create partitions for Linux.

DR-DOS 5.0 and 6.0 are reported to have difficulties with partition ID codes of 80 or more. The Linux '`fdisk`' used to set the system type of new partitions to hexadecimal 81. DR-DOS seems to confuse this with hexadecimal 1, a DOS code. The values 82 for swap and 83 for file systems should not cause problems with DR-DOS. If they do, you may use the '`fdisk`' command '`t`' to change the system code of any Linux partitions to some number less than hexadecimal 80; I suggest 42 and 43 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.

6.3. 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. (I counted 36 extensions in `linux/fs/msdos/misc.c` which are treated as marking binary files. All other files are treated as text.)

6.4. 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 then *mtools* provide 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`	put DOS file system on a formatted disk
`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

Note that the '`mformat`' command does not do a low-level format of the disk. For this you must use the command '`fdformat`'.

The above 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. If you wish to use '`mformat`', you must specify additional entries in the `/etc/mtools` configuration file for the floppy formats you wish to be able to write. These are the entries which I use to format 1.4 Mb floppies in drive 0 and 1.2 Mb floppies in drive 1:

```
E /dev/fd0 12 80 2 18
F /dev/fd1 12 80 2 15
```

Then I can low-level format a floppy using '`fdformat`', and add an *msdos* file system with the command '`mformat E:`' or '`mformat F:`'.

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.

6.5. 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:

```
badblocks(8)  fsck.ext2(8)   mkfs(8)          setfdprm(8)
clock(8)      fsck.minix(8)  mkfs.ext2(8)     setfont(8)
doshell(8)    fstab(5)       mkfs.minix(8)    setserial(8)
dumpe2fs(8)   halt(8)        mklost+found(8)  showfont(8)
dumpkeys(1)   init(8)        mkswap(8)        shutdown(8)
fdformat(8)   kbdrate(8)     mount(8)         swapon(8)
frag(8)       killall(1)     rdev(8)          tune2fs(8)
fsck(8)       mapscrn(8)     selection(1)     update(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. Most of the manual pages in sections 4 and 5 contain interesting information. For a list, give the command '`ls /usr/man/cat[45]`'.

6.6. 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 `/sbin/init` or one of the shared libraries in `/lib`.

In some 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.conf`; 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. In the simplest cases, boot from the MCC boot floppy, and give the command 'ro root=*xxx*' at the LILO prompt.

If that does not work, boot from the MCC boot floppy, giving the command 'ro' at the LILO prompt. Place the root floppy in the drive when you are asked to do so. 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 MCC floppies. As with all MCC upgrades, this creates a special directory named /backupdirs, into 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. You will need to install LILO boot code again before rebooting. Use the command

```
ROOT=/root;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 /boot/vmlinuz 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 /boot/vmlinuz /mnt
tar cf - /etc/psdatabase|gzip -9 >/mnt/psdata.tgz
lilo -C /mnt/lilo.conf
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.

7. Available Documentation for Linux

The documentation available for Linux can be divided into online documentation, collections of Frequently Asked Questions (with answers), Linux HOWTO documents, the standard for the structure of Linux file systems, and other sources of information.

7.1. Documentation available on line

There are at least four kinds of information available online, at least if you have installed all of the MCC Interim packages:

internal help
: Many Linux commands contain helpful information of some kind. Some, like the `bash` shell, contain an interactive '`help`' command. Others, like the `tar` command, give help when certain command line arguments are given, e.g., '`--help`' or '`--wxyz`'.

online manuals
: The system of online manuals is accessed using the '`man`' command. Type '`man man`' for more information. In the course of time, more and more manuals are being written or adapted for Linux.

info files
: The *info* system consists of specially formatted manuals which can be read using the '`info`' or '`emacs`' command. These manuals have a complex tree structure, and special commands allow you to traverse the tree, use the indices, search for strings, and even jump to other manuals in the info system.

text files
: I have included in the distribution a few files describing commands which are otherwise poorly documented on line. For example, the '`lilo`' command has no other information on line, so I have placed the file `README.lilo` in `/etc`.

7.2. Collections of Frequently Asked Questions

Collections of frequently asked questions, with answers, are a traditional form of information in the Unix and Usenet communities. Many of these can be found at `ftp.mcc.ac.uk` in the directory `/pub/linux/fi.mirror/doc/FAQ`.

The mail programs in this distribution, '`elm`' and '`smail`', both have informative FAQs. The file `/usr/lib/emacs/19.22/etc/FAQ` is contained in the MCC *emacs* package.

FAQ files not peculiar to Linux can be found at `rtfm.mit.edu` in a very large directory tree. Other sites which mirror this tree are `ftp.uu.net` in `/usenet/news.answers`, `grasp1.univ.lyon1.fr` in `/pub/faq`, and `ftp.win.tue.nl` in `/pub/usenet/news.answers`.

7.3. The Linux HOWTO documents

The Linux community seems to have invented a new kind of document, more organised (in most cases) than the traditional FAQ file. This is the HOWTO file, most of which can be found at `ftp.mcc.ac.uk` in the directory `/pub/linux/fi.mirror/doc/HOWTO`.

Of particular interest in this category are the HOWTO Index, which lists the currently available HOWTO files, as well as HOWTO files for supported hardware, QIC ftape drivers, mail, Net-2, network news, printing, SCSI devices, Serial ports, sound hardware and software, '`uucp`', and XFree86.

Many of these documents have not yet been updated to take into account the new file system standard. This means that the full path names of many files and directories will not be the same as those which you will install on your system as parts of the MCC distribution.

7.4. The Linux file system standard

This and other releases of Linux are heavily influenced by the new standard for the Linux directory structure, *Linux Filesystem Structure*. This replaces the long-outdated standard which most distributions have more or less ignored in the past year.

This MCC Interim release tries to observe the new standard as closely as possible. It diverges from the published standard (version 1.0) in the following respects:

- Several draft standards have been circulated privately since the official release of version 1.0. MCC Interim Linux follows the March 18th draft where it differs from the earlier document.
- The standard, which is still evolving, has not yet specified the location of many portions of packages such as TEX, newsreaders, and other software. Moreover, some software which is not distributed as part of the MCC package, is not yet available for installation in conforming packages. The presence of paths such as `/usr/TeX/bin` in files such as `/etc/profile` reflects this.
- The standard has specified that C++ include files should not be in `/usr/g++-include`, but in `/usr/include/g++`. This distribution does not feel free to change that location until after H J Lu has adopted this convention in his 'official' distribution of the '`g++`' support files.
- The standard is, in my view, unclear about the location of keyboard map files. I have placed them in `/usr/lib/keytables/keytables` rather than in `/etc/keytables/keytables`. The default map file is in `/etc/kbmap`, which was used by previous MCC distributions. The standard will probably specify another name for the default keyboard map file.
- The standard specifies that the file `/usr/dict/words` should be required in any fully compliant implementation. This file is not contained in the MCC Interim distribution. A copy of a `words` file is available by anonymous ftp from `ftp.mcc.ac.uk` in `/pub/linux/binaries-4.5.21/usr/dict/words`. A contributed package which installs this file is available from `ftp.mcc.ac.uk` in `/bin/linux/mcc-interim/1.0+/contributions`.
- The standard specifies that `/usr/include/asm` and `/usr/include/linux` should be symbolic links to subdirectories of `/usr/src/linux/include`, but in this distribution the actual files

are installed in `/usr/include`. This document contains instructions for moving the files to the standard location. See Section 5.1 [Recompiling the kernel], page 41.

In the terms defined in the March 18 draft of the Linux file system standard, I believe that the MCC Interim version is 'fully compatible' with the standard, and that only the absence of `/usr/dict/words` and the location of the files in `/usr/g++-include`, `/usr/include/asm`, and `/usr/include/linux` prevent it from being 'fully compliant'.

7.5. Other information about Linux

The Linux Documentation Project has produced four largish volumes, which are available by anonymous ftp and in a commercially printed form. Newcomers should note particularly *Linux Installation and Getting Started*, by Matt Welsh. This contains a lot of useful information, as well as pointers to other sources of information. It is written in a clear but conversational style, and so may be particularly helpful to beginners. The book contains discussions of basic commands, including shell commands, pipes and redirection, file permissions, job control, the '`vi`' editor, shell scripts, system administration, and many other topics. It also contains a list of recommended books about Unix.

The *Linux System Administrator's Guide*, by Lars Wirzenius, is also quite useful and informative. The present version is older than the Linux file system standard, and much information is missing or not yet written. These places are marked with the word '**META**' in bold type. But the booklet contains much useful information that is otherwise difficult to find.

For these and other works from the Linux Documentation Project, see `ftp.mcc.ac.uk` in the directory `/pub/linux/fi.mirror/doc/doc-project`.

Another source of information about Linux is the USENET news service, where there are a growing number of `comp.os.linux` groups. There are also several mailing lists related to Linux. For further information about mailing lists and USENET news groups which are relevant to Linux, see *Linux Installation and Getting Started*.

Appendix A. Acknowledgments

This distribution of Linux owes a great deal to the testers who have helped find various problems before the system was released, and to all those who have reported difficulties and sent in patches.

The following information acknowledges only the primary author of each command. In many cases a program has been largely rewritten or altered by a number of contributors; you must consult the source for more information. Please note in particular the file `/usr/src/linux/CREDITS`, which contains the names of many programmers who have contributed to the development of the Linux kernel.

The suffix `.tgz` is omitted from the names of the package and source files. The following commands are installed by MCC Interim packages:

Command	Directory	Package	Source	Author(s)
[	/usr/bin	base2	shelu194	GNU
addftinfo	/usr/bin	groff	groff109	James Clark
afmtodit	/usr/bin	groff	groff109	James Clark
apropos	/usr/bin	base1	man11a	John Eaton
ar	/usr/bin	gcca	binu1914	GNU
arch	/bin	base1	utilx15	Rickard E Faith
as	/usr/bin	gcca	gas22	GNU
as86	/usr/bin	gcca	bin8601	Bruce Evans
awk	/usr/bin	gawk	gawk2154	GNU
badblocks	/sbin	base1	e2fscm15	Remy Card
banner	/usr/bin	base2	utilx15	BSD
basename	/usr/bin	base1	shelu194	GNU
bash	/bin	base1	bash1135	Brian Fox, GNU
bison	/usr/bin	bison	bison122	GNU
brik	/usr/bin	base1	brik155	Rahul Dhesi
c++	/usr/bin	gpp	gcc258	GNU
cal	/usr/bin	base1	utilx15	BSD
cat	/bin	base1	txtu19	GNU
cc	/usr/bin	gcca	gcc258	GNU
chattr	/usr/bin	base2	e2fscm15	Remy Card
checkalias	/usr/bin	mail	elm24	Dave Taylor
chgrp	/bin	base1	filu39	GNU
chmod	/bin	base1	filu39	GNU
chown	/bin	base1	filu39	GNU
chroot	/usr/sbin	base1	utilx15	Rick Sladkey
chsh	/usr/bin	base1	utilx15	Peter Orbaek
cksum	/usr/bin	base1	txtu19	GNU
clock	/sbin	base1	utilx15	Charles Hedrick
cmp	/usr/bin	base1	diffu26	T Granlund, D MacKenzie
col	/usr/bin	base2	utilx15	BSD
colcrt	/usr/bin	base2	utilx15	BSD
colrm	/usr/bin	base2	utilx15	BSD

column	/usr/bin	base2	utilx15	BSD
comm	/usr/bin	base1	txtu19	GNU
compress	/usr/bin	base1	comp401	Joseph M Orost
cp	/bin	base1	filu39	GNU
cpio	/usr/bin	base1	cpio23	GNU
cpp	/usr/bin	gccb	gcc258	GNU
csplit	/usr/bin	base1	txtu19	GNU
ctags	/usr/bin	base1	elvi17	Steve Kirkendall
ctrlaltdel	/usr/sbin	base1	utilx15	Peter Orbaek
cut	/usr/bin	base1	txtu19	GNU
date	/bin	base1	shelu194	GNU
dd	/bin	base1	filu39	GNU
debugfs	/usr/sbin	base2	e2fscm15	Theodore Ts'o
df	/bin	base1	filu39	GNU
diff	/usr/bin	base2	diffu26	Mike Haertel et al
diff3	/usr/bin	base2	diffu26	Randy Smith
dir	/bin	base1	filu39	GNU
dirdump	/usr/sbin	base1	e2fsp04a	Remy Card
dirname	/usr/bin	base1	shelu194	GNU
dmesg	/bin	base1	utilx15	Theodore Ts'o
domainname	/bin	tcpip	utilx15	Peter Orbaek
doshell	/usr/bin	base1	utilx15	Jim Wiegand
du	/usr/bin	base1	filu39	GNU
dumpe2fs	/usr/sbin	base2	e2fscm15	Remy Card
dumpkeys	/usr/bin	base1	kbd085	Risto Kankkunen
echo	/bin	base1	shelu194	GNU
ed	/bin	base2	utilx15	BSD/Kernighan/Plauger
egrep	/usr/bin	base1	grep20	GNU
elm	/usr/bin	mail	elm24	Dave Taylor
elmalias	/usr/bin	mail	elm24	Dave Taylor
elvis	/usr/bin	base2	elvi17	Steve Kirkendall
elvprsv	/usr/bin	base1	elvi17	Steve Kirkendall
elvrec	/usr/bin	base1	elvi17	Steve Kirkendall
emacs	/usr/bin	emacs	emac1922	Richard M Stallman
emacs-19.22	/usr/bin	emacs	emac1922	Richard M Stallman
emacsclient	/usr/bin	emacs	emac1922	Richard M Stallman
env	/usr/bin	base1	shelu194	GNU
etags	/usr/bin	emacs	emac1922	Richard M Stallman
ex	/usr/bin	base2	elvi17	Steve Kirkendall
expand	/usr/bin	base1	txtu19	GNU
expr	/usr/bin	base1	shelu194	GNU
false	/bin	base1	shelu194	GNU
fastmail	/usr/bin	mail	elm24	Dave Taylor
fdformat	/usr/bin	base1	utilx15	Werner Almesberger
fdisk	/sbin	base1	fdisk15	A V Le Blanc
fgrep	/usr/bin	base1	grep20	GNU
file	/usr/bin	base1	file	Ian F Darwin
filter	/usr/bin	mail	elm24	Dave Taylor

find	/usr/bin	base1	find38	GNU
finger	/usr/bin	tcpip	netsr25m	BSD
fingerd	/usr/sbin	tcpip	netsr25m	BSD
flex	/usr/bin	flex	flex246	Vern Paxson
flex++	/usr/bin	flex	flex246	Vern Paxson
fmt	/usr/bin	base1	elvi17	Steve Kirkendall
fold	/usr/bin	base1	txtu19	GNU
frag	/usr/sbin	base1	utilx15	Werner Almesberger
free	/usr/bin	base1	procps08	Brian Edmonds
frm	/usr/bin	mail	elm24	Dave Taylor
fsck	/sbin	base1	utilx15	David Engel
fsck.ext2	/sbin	base1	e2fscm15	Theodore Ts'o
fsck.minix	/sbin	base1	utilx15	Linus Torvalds
ftp	/usr/bin	tcpip	netsr25m	BSD
ftpd	/usr/sbin	tcpip	netsr25m	BSD
fuser	/usr/bin	base1	procps08	Werner Almesberger
g++	/usr/bin	gpp	gcc258	GNU
gawk	/usr/bin	gawk	gawk2154	GNU
gcc	/usr/bin	gcca	gcc258	GNU
gdb	/usr/bin	gdb	gdb412	GNU
genclass	/usr/bin	gpp	lg++253	GNU
geqn	/usr/bin	groff	groff109	James Clark
getty	/sbin	base1	gtyp207b	Paul Sutcliffe Jr
gindxbib	/usr/bin	groff	groff109	James Clark
glookbib	/usr/bin	groff	groff109	James Clark
gneqn	/usr/bin	groff	groff109	James Clark
gnroff	/usr/bin	groff	groff109	James Clark
gpic	/usr/bin	groff	groff109	James Clark
gprof	/usr/bin	gprof	binu1914	GNU
grefer	/usr/bin	groff	groff109	James Clark
grep	/usr/bin	base1	grep20	GNU
grodvi	/usr/bin	groff	groff109	James Clark
groff	/usr/bin	groff	groff109	James Clark
grog	/usr/bin	groff	groff109	James Clark
grops	/usr/bin	groff	groff109	James Clark
grotty	/usr/bin	groff	groff109	James Clark
groups	/usr/bin	base1	shelu194	GNU
gsoelim	/usr/bin	groff	groff109	James Clark
gtbl	/usr/bin	groff	groff109	James Clark
gtroff	/usr/bin	groff	groff109	James Clark
gunzip	/usr/bin	base1	gzip124	Jean-loup Gailly
gzexe	/usr/bin	base2	gzip124	Jean-loup Gailly
gzip	/usr/bin	base1	gzip124	Jean-loup Gailly
halt	/sbin	base1	svini250	Miquel van Smoorenburg
head	/usr/bin	base1	txtu19	GNU
hexdump	/usr/bin	base2	utilx15	BSD
hostname	/bin	base1	mccmisc	Fred N van Kempen
id	/usr/bin	base1	shelu194	GNU

ifconfig	/sbin	tcpip	netsr25m	Fred N van Kempen
iflink	/sbin	tcpip	netsr25m	Fred N van Kempen
ifsetup	/sbin	tcpip	netsr25m	Fred N van Kempen
inetd	/usr/sbin	tcpip	netsr25m	BSD
info	/usr/bin	info	texi31	Brian Fox
init	/sbin	base1	svini250	Miquel van Smoorenburg
install	/usr/bin	base1	filu39	GNU
install.lilo	/usr/sbin	base2	-	A V Le Blanc
install.mail	/usr/sbin	mail	-	Ian Kluft
install.net	/usr/sbin	tcpip	-	A V Le Blanc
ipcrm	/usr/bin	base1	utilx15	Krishna Balasubramanian
ipcs	/usr/bin	base1	utilx15	Krishna Balasubramanian?
joe	/usr/bin	base2	joe108	Joseph H Allen
join	/usr/bin	base1	txtu19	GNU
kbdinstall	/usr/sbin	base2	mccmisc	A V Le Blanc
kbdrate	/sbin	base1	utilx15	Rickard E Faith
kermit	/usr/bin	kermit	ckrmu189	Frank da Cruz
kill	/bin	base1	utilx15	Peter MacDonald
killall	/bin	base1	procps08	Werner Almesberger
last	/usr/bin	base1	svini250	Miquel van Smoorenburg
ld	/usr/bin	gcca	binu1914	GNU
ld86	/usr/bin	gcca	bin8601	Bruce Evans
ldconfig	/usr/sbin	base1	ldso143	David Engel, Mitch D'Souza
ldd	/usr/bin	base1	ldso143	David Engel
less	/usr/bin	base2	less177	Mark Nudelman
lesskey	/usr/bin	base2	less177	Mark Nudelman
lilo	/sbin	base1	lilo14	Werner Almesberger
listalias	/usr/bin	mail	elm24	Dave Taylor
lkbib	/usr/bin	groff	groff109	James Clark
ll	/bin	base1	filu39	GNU
ln	/bin	base1	filu39	GNU
loadkeys	/bin	base1	kbd085	Risto Kankkunen
locate	/usr/bin	base1	find38	GNU
logger	/usr/bin	base1	utilx15	BSD
login	/bin	base1	utilx15	BSD
logname	/usr/bin	base1	shelu194	GNU
look	/usr/bin	base1	utilx15	Rickard E Faith
lpc	/usr/bin	lp	netsr25m	BSD
lpd	/usr/sbin	lp	netsr25m	BSD
lpf	/usr/sbin	lp	netsr25m	BSD
lpq	/usr/bin	lp	netsr25m	BSD
lpr	/usr/bin	lp	netsr25m	BSD
lprm	/usr/bin	lp	netsr25m	BSD
lptest	/usr/bin	lp	netsr25m	BSD
ls	/bin	base1	filu39	GNU
lsattr	/usr/bin	base2	e2fscm15	Remy Card
lsf	/bin	base1	filu39	GNU
mail	/usr/bin	mail	smai3128	M Horton, L C Noll

mailq	/usr/bin	mail	smai3128	R S Karr, L C Noll
make	/usr/bin	gcca	make370	GNU
makehole	/usr/sbin	base1	mccmisc	H J Lu
makeinfo	/usr/bin	info	texi31	Brian Fox
makewhatis	/usr/sbin	base1	man11a	John Eaton
man	/usr/bin	base2	man11a	John Eaton
manpath	/usr/bin	base2	man11a	John Eaton
mapscrn	/usr/bin	base1	kbd085	Andries Brouwer
mattrib	/usr/bin	base2	mtoln2	Emmet P Gray
mccinstall	/usr/sbin	base1	-	A V Le Blanc
mcd	/usr/bin	base2	mtoln2	Emmet P Gray
mcopy	/usr/bin	base2	mtoln2	Emmet P Gray
mdel	/usr/bin	base2	mtoln2	Emmet P Gray
mdir	/usr/bin	base2	mtoln2	Emmet P Gray
mesg	/usr/bin	base1	svini250	Miquel van Smoorenburg
messages	/usr/bin	mail	elm24	Dave Taylor
mformat	/usr/bin	base2	mtoln2	Emmet P Gray
mk_modmap	/usr/bin	base1	kbd085	Kjetil T Homme
mkaliases	/usr/bin	mail	smai3128	R S Karr, L C Noll
mkdir	/bin	base1	filu39	GNU
mkfifo	/usr/bin	base1	filu39	GNU
mkfs	/sbin	base1	utilx15	David Engel
mkfs.ext2	/sbin	base1	e2fscm15	Theodore Ts'o
mkfs.minix	/sbin	base1	utilx15	Linus Torvalds
mklost+found	/usr/sbin	base1	e2fscm15	Remy Card
mkmanifest	/usr/bin	base2	mtoln2	Emmet P Gray
mknod	/bin	base1	filu39	GNU
mksuper	/usr/bin	base1	e2fsp04a	Remy Card
mkswap	/sbin	base1	utilx15	Linus Torvalds
mlabel	/usr/bin	base2	mtoln2	Emmet P Gray
mmd	/usr/bin	base2	mtoln2	Emmet P Gray
more	/bin	base1	utilx15	BSD
mount	/bin	base1	utilx15	Doug Quale
mrd	/usr/bin	base2	mtoln2	Emmet P Gray
mread	/usr/bin	base2	mtoln2	Emmet P Gray
mren	/usr/bin	base2	mtoln2	Emmet P Gray
mt	/usr/bin	base2	cpio23	GNU
mtype	/usr/bin	base2	mtoln2	Emmet P Gray
mv	/bin	base1	filu39	GNU
mwrite	/usr/bin	base2	mtoln2	Emmet P Gray
neqn	/usr/bin	groff	groff109	James Clark
netstat	/bin	tcpip	netsr25m	Fred Baumgarten
newalias	/usr/bin	mail	elm24	Dave Taylor
newaliases	/usr/bin	mail	smai3128	R S Karr, L C Noll
newgrp	/usr/bin	base1	utilx15	Michael Haardt
newmail	/usr/bin	mail	elm24	Dave Taylor
nfrm	/usr/bin	mail	elm24	Dave Taylor
nice	/usr/bin	base1	shelu194	GNU

nl	/usr/bin	base1	txtu19	GNU
nm	/usr/bin	gcca	binu1914	GNU
nohup	/usr/bin	base1	shelu194	GNU
nroff	/usr/bin	groff	groff109	James Clark
nslookup	/usr/bin	tcpip	netsr25m	BSD
nsquery	/usr/bin	tcpip	netsr25m	BSD
ntalkd	/usr/sbin	tcpip	netsr25m	BSD
objdump	/usr/bin	gcca	binu1914	GNU
od	/usr/bin	base2	txtu19	GNU
passwd	/usr/bin	base1	utilx15	Peter Orbaek
paste	/usr/bin	base1	txtu19	GNU
patch	/usr/bin	base2	patch21	Larry Wall
pathchk	/usr/bin	base1	shelu194	GNU
pathto	/usr/bin	mail	smai3128	R S Karr, L C Noll
pfbtops	/usr/bin	groff	groff109	James Clark
ping	/bin	tcpip	netsr25m	BSD
powerd	/sbin	base1	svini250	Miquel van Smoorenburg
pr	/usr/bin	base2	txtu19	GNU
printenv	/usr/bin	base1	shelu194	GNU
printf	/usr/bin	base1	shelu194	GNU
printmail	/usr/bin	mail	elm24	Dave Taylor
protoize	/usr/bin	gcca	gcc258	GNU
ps	/bin	base1	procps08	Branko Lankester
psbb	/usr/bin	groff	groff109	James Clark
pstree	/usr/bin	base1	procps08	Werner Almesberger
psupdate	/usr/sbin	base1	procps08	Branko Lankester
pwd	/bin	base1	utilx15	BSD
ramsize	/usr/bin	base1	utilx15	Werner Almesberger
ranlib	/usr/bin	gcca	binu1914	GNU
rcp	/usr/bin	tcpip	netsr25m	BSD
rdev	/usr/bin	base1	utilx15	Werner Almesberger
readmsg	/usr/bin	mail	elm24	Dave Taylor
reboot	/sbin	base1	svini250	Miquel van Smoorenburg
ref	/usr/bin	base1	elvi17	Steve Kirkendall
renice	/usr/bin	base1	utilx15	BSD
rev	/usr/bin	base1	utilx15	BSD
rlogin	/usr/bin	tcpip	netsr25m	BSD
rlogind	/usr/sbin	tcpip	netsr25m	BSD
rm	/bin	base1	filu39	GNU
rmail	/usr/bin	mail	smai3128	R S Karr, L C Noll
rmdir	/bin	base1	filu39	GNU
rootflags	/usr/bin	base1	utilx15	Werner Almesberger
route	/sbin	tcpip	netsr25m	Fred N van Kempen
rsh	/usr/bin	tcpip	netsr25m	BSD
rshd	/usr/sbin	tcpip	netsr25m	BSD
rsmtp	/usr/bin	mail	smai3128	R S Karr, L C Noll
runlevel	/usr/bin	base1	svini250	Miquel van Smoorenburg
runq	/usr/bin	mail	smai3128	R S Karr, L C Noll

script	/usr/bin	base1	utilx15	BSD
sdiff	/usr/bin	base2	diffu26	Thomas Lord
sed	/usr/bin	base1	sed203	GNU
selection	/usr/bin	base2	utilx15	Andrew Haylett
sendmail	/usr/sbin	mail	smai3128	R S Karr, L C Noll
setfdprm	/usr/bin	base1	utilx15	Werner Almesberger
setfont	/usr/bin	base1	kbd085	E Crosser, A Brouwer
setleds	/usr/bin	base1	kbd085	Andries Brouwer
setmetamode	/usr/bin	base1	kbd085	Andries Brouwer
setserial	/bin	base1	setsr202	Michael K Johnson
setsid	/usr/bin	base1	utilx15	Rick Sladkey
setterm	/usr/bin	base1	utilx15	Gordon Irlam
sh	/bin	base1	bash1135	Brian Fox, GNU
showfont	/usr/bin	base1	kbd085	Andries Brouwer
showkey	/usr/bin	base1	kbd085	Risto Kankkunen
shutdown	/sbin	base1	svini250	Miquel van Smoorenburg
size	/usr/bin	gcca	binu1914	GNU
sleep	/usr/bin	base1	shelu194	GNU
smail	/usr/bin	mail	smai3128	R S Karr, L C Noll
smtpd	/usr/bin	mail	smai3128	R S Karr, L C Noll
sort	/usr/bin	base1	txtu19	GNU
split	/usr/bin	base1	txtu19	GNU
strings	/usr/bin	base1	utilx15	BSD
strip	/usr/bin	gcca	binu1914	GNU
stty	/bin	base1	shelu194	GNU
su	/bin	base1	shelu194	GNU
sum	/usr/bin	base1	txtu19	GNU
swapdev	/usr/bin	base1	utilx15	Werner Almesberger
swapoff	/sbin	base1	utilx15	Doug Quale
swapon	/sbin	base1	utilx15	Doug Quale
sync	/bin	base1	utilx15	Linus Torvalds
syslogd	/usr/sbin	base1	utilx15	BSD
tac	/usr/bin	base1	txtu19	GNU
tail	/usr/bin	base1	txtu19	GNU
talk	/usr/bin	tcpip	netsr25m	BSD
tar	/usr/bin	base2	tar1112	GNU
tbl	/usr/bin	groff	groff109	James Clark
tee	/usr/bin	base1	shelu194	GNU
telinit	/sbin	base1	svini250	Miquel van Smoorenburg
telnet	/usr/bin	tcpip	netsr25m	BSD
telnetd	/usr/sbin	tcpip	netsr25m	BSD
test	/usr/bin	base2	shelu194	GNU
tfmtodit	/usr/bin	groff	groff109	James Clark
time	/usr/bin	base1	time16	GNU
tload	/usr/bin	base1	procps08	Branko Lankester
top	/usr/bin	base1	procps08	Roger Binns
touch	/usr/bin	base1	filu39	GNU
tr	/usr/bin	base1	txtu19	GNU

true	/bin	base1	shelu194	GNU
tsort	/usr/bin	base1	utilx15	BSD
tty	/usr/bin	base1	shelu194	GNU
tune2fs	/sbin	base1	e2fscm15	Remy Card
ul	/usr/bin	base1	utilx15	BSD
umount	/bin	base1	utilx15	Doug Quale
uname	/bin	base1	shelu194	GNU
uncompress	/usr/bin	base1	comp401	Joseph M Orost
unexpand	/usr/bin	base1	txtu19	GNU
uniq	/usr/bin	base1	txtu19	GNU
unprotoize	/usr/bin	gcca	gcc258	GNU
update	/sbin	base1	utilx15	Linus Torvalds
updatedb	/usr/sbin	base1	find38	GNU
uptime	/usr/bin	base1	procps08	Michael K Johnson
users	/usr/bin	base1	-	?
utmpdump	/usr/sbin	base1	svini250	Miquel van Smoorenburg
uudecode	/usr/bin	base1	uuenc10	GNU, BSD
uuencode	/usr/bin	base1	uuenc10	GNU, BSD
uugetty	/sbin	base2	gtyp207b	Paul Sutcliffe Jr
uupath	/usr/bin	mail	smai3128	R S Karr, L C Noll
uuwho	/usr/bin	mail	smai3128	Gordon Moffett
vdir	/bin	base1	filu39	GNU
vi	/usr/bin	base2	elvi17	Steve Kirkendall
vidmode	/usr/bin	base1	utilx15	Werner Almesberger
virec	/usr/bin	base1	elvi17	Steve Kirkendall
vlock	/usr/bin	base2	vlock06	Michael K Johnson
vmlinuz	/boot	base2	lx10	Linus Torvalds
w	/usr/bin	base2	procps08	Larry Greenfield
wall	/usr/bin	base1	svini250	Miquel van Smoorenburg
wc	/usr/bin	base1	txtu19	GNU
wdsetup	/sbin	tcpip	netsr25m	Gregg Weber
whatis	/usr/bin	base1	man11a	John Eaton
whereis	/usr/bin	base1	utilx15	BSD
who	/usr/bin	base1	shelu194	GNU
whoami	/usr/bin	base1	shelu194	GNU
wnewmail	/usr/bin	mail	elm24	Dave Taylor
write	/usr/bin	base1	utilx15	BSD
xargs	/usr/bin	base1	find38	GNU
yes	/usr/bin	base1	shelu194	GNU
zcat	/usr/bin	base1	gzip124	Jean-loup Gailly
zcmp	/usr/bin	base2	gzip124	Jean-loup Gailly
zdiff	/usr/bin	base2	gzip124	Jean-loup Gailly
zdump	/usr/bin	timezone	utilx15	?
zforce	/usr/bin	base1	gzip124	Jean-loup Gailly
zgrep	/usr/bin	base2	gzip124	Jean-loup Gailly
zic	/usr/bin	timezone	utilx15	?
zmore	/usr/bin	base1	gzip124	Jean-loup Gailly
znew	/usr/bin	base1	gzip124	Jean-loup Gailly

File Index

A

adm 49
adm-config 48

B

backupdirs 4, 64
bin/sh 64
boot.xxxx 37
boot/boot.xxxx 37
boot/vmlinuz 36
boot/vmlinuz.old 36, 41
bootinstall 32
Bugs+Warnings 5, 9

C

cdrmboot.cnf 9
cdrmboot.gz 9
config.in 9
CREDITS 69

D

dev/hda 23
dev/hdb 23
dev/mouse 44
dev/sda 23
dev/sdb 23
dev/xda 23
dev/xdb 23
disktab 37, 45

E

etc/disktab 37, 45
etc/fstab 17
etc/host.conf 44
etc/inetd.conf 43
etc/init 63
etc/inittab 4
etc/kbmap 30, 45
etc/lilo.conf 35, 36, 44
etc/mtools 61
etc/passwd 64
etc/rc 4, 52
etc/rc.local 44
etc/README.lilo 35
etc/resolv.conf 44

F

filename.crc 12
fstab 17

H

hdb 23
host.conf 44

I

include/asm 41
include/linux 41
inetd.conf 43
init 63
inittab 4
install.clean 12
install.info 12
install.setup 12
ipide.cnf 10

K

kbmap 30, 45

L

libbfd.a 10
libdbm.a 10
libgmon.a 10
libiberty.a 10
libmcheck.a 10
libmmalloc.a 10
libopcodes.a 10
libreadline.a 10
lilo 35
lilo.conf 35, 36, 44
linux-mcc/install 36
linux/init/main.c 46

M

main.c 46
messages 49
mouse 44
mtools 61

N

nocdboot.cnf 9
nocdboot.gz 9

P

passwd 64
pwd.h 44

R

rc 4, 52
rc.local 44
README files 9
README.Config 47
README.lilo 35
resolv.conf 44
root.gz 9

S

sbin/lilo 35
sda 23
sdb 23
sh 64

T

tmp/bootinstall 32

U

usr/dict/words 10, 66
usr/include/asm 41
usr/include/linux 41
usr/include/pwd.h 44
usr/lib/libbfd.a 10
usr/lib/libdbm.a 10
usr/lib/libgmon.a 10
usr/lib/libiberty.a 10
usr/lib/libmcheck.a 10
usr/lib/libmmalloc.a 10
usr/lib/libopcodes.a 10
usr/lib/libreadline.a 10
usr/lib/X11/etc/README.Config 47
usr/lib/X11/xdm/adm-config 48
usr/lib/X11/xdm/Xsetup_0 48
usr/src/linux-mcc/install 36
usr/src/linux/CREDITS 69
usr/src/linux/init/main.c 46
usr/X386/lib/X11/Xconfig 47

V

var/adm 49
var/adm/messages 49
var/adm/wtmp 49
vmlinuz 36
vmlinuz.old 36, 41

W

wd.cnf 10
words 10, 66
wtmp 49

X

Xconfig 47
xda 23
xdb 23
Xsetup_0 48

Program Index

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

[

[..... 69

A

addftinfo 69
afmtodit 69
apropos 69
ar 69
arch 69
as 10, 69
as86 69
awk 69

B

badblocks 69
banner 69
basename 69
bash 65, 69
bison 69
bootinstall 32
brik 69
  verifying MCC packages 12, 29

C

c++ 69
cal 69
cat 69
cc 69
chattr 69
checkalias 69
chgrp 69
chmod 49, 69
chown 69
chroot 69
chsh 69
cksum 69
clock 69
cmp 69
col 69
colcrt 69
colrm 69
column 70
comm 70
compress 70
cp 70
cpio 70
cpp 70
csplit 70
ctags 70
ctrlaltdel 70
cut 70

D

date 70
dd 70
debugfs 70
df 70
diff 70
diff3 70
dir 70
dirdump 70
dirname 70
dmesg 70
domainname 70
doshell 70
du 70
dumpe2fs 70
dumpkeys 70

E

echo 70
ed 70
egrep 70
elm 10, 65, 70
elmalias 70
elvis 70
elvprsv 70
elvrec 70
emacs 10, 65, 70
emacs-19.22 70
emacsclient 70
env 70
etags 70
ex 70
expand 70
expr 70
EZSETUP 42

F

false 70

fastmail 70
fdformat 70
fdisk 53, 70
  description 50
  from MCC root floppy 23
  reboot after running 25
fgrep 70
file 70
filter 70
find 71
finger 71
fingerd 71
flex 71
flex++ 71
fmt 71
fold 71
frag 71
free 71
frm 71
fsck 27, 49, 71
fsck.ext2 71
fsck.minix 71
ftp 71
ftpd 71
fuser 71

G

g++ 71
gas 10
gawk 71
gcc 71
gdb 71
genclass 71
geqn 71
getty 71
gindxbib 71
glookbib 71
gneqn 71
gnroff 71
gpic 71
gprof 10, 71
grefer 71
grep 71
grodvi 71
groff 71
grog 71
grops 71
grotty 71
groups 71
gsoelim 71
gtbl 71
gtroff 71
gunzip 71
gzexe 71
gzip 71
GZIP.EXE 11
GZIPxxx.EXE 11

H

halt 71
head 71
hexdump 71
hostname 71

I

id 71
ifconfig 72
iflink 42, 72
ifsetup 42, 72
inetd 72
info 65, 72
init 72
install 72
install.lilo 32, 35, 72
install.mail 72
install.net 31, 33, 43, 72
ipcrm 72
ipcs 72

J

joe 72
join 72

K

kbdinstall 45, 72
kbdrate 72
kermit 10, 72
kill 72
killall 72

L

last 72
ld 72
ld86 72
ldconfig 72
ldd 72
less 72
lesskey 72
lilo 35, 72
listalias 72
lkbib 72
ll 72
ln 72
loadkeys 72

locate ... 72
logger ... 72
login ... 72
logname ... 72
look ... 72
lpc ... 72
lpd ... 72
lpf ... 72
lpq ... 72
lpr ... 10, 72
lprm ... 72
lptest ... 72
ls ... 72
lsattr ... 72
lsf ... 72

M

mail ... 72
mailq ... 73
make ... 73
makehole ... 29, 73
makeinfo ... 73
makewhatis ... 73
man ... 65, 73
manpath ... 73
mapscrn ... 73
mattrib ... 61, 73
mccinstall ... 33, 73
mcd ... 61, 73
mcopy ... 61, 73
mdel ... 61, 73
mdir ... 61, 73
mesg ... 73
messages ... 73
mformat ... 61, 73
mk_modmap ... 73
mkaliases ... 73
mkdir ... 73
mkfifo ... 73
mkfs ... 73
mkfs.ext2 ... 73
mkfs.minix ... 73
mklost+found ... 73
mkmanifest ... 73
mknod ... 73
mksuper ... 73
mkswap ... 17, 73
mlabel ... 61, 73
mmd ... 61, 73
more ... 73
mount ... 60, 73
mrd ... 61, 73
mread ... 61, 73
mren ... 61, 73
mt ... 73
mtype ... 61, 73
mv ... 73
mwrite ... 61, 73

N

neqn ... 73
netstat ... 73
newalias ... 73
newaliases ... 73
newgrp ... 73
newmail ... 73
nfrm ... 73
nice ... 73
nl ... 74
nm ... 74
nohup ... 74
nroff ... 74
nslookup ... 74
nsquery ... 74
ntalkd ... 74

O

objdump ... 74
od ... 74

P

passwd ... 49, 74
paste ... 74
patch ... 74
pathchk ... 74
pathto ... 74
pfbtops ... 74
ping ... 74
powerd ... 74
pr ... 74
printenv ... 74
printf ... 74
printmail ... 74
protoize ... 74
ps ... 74
psbb ... 74
pstree ... 74
psupdate ... 41, 74
pwd ... 74

R

ramsize ... 74
ranlib ... 74
RAWRITE.EXE ... 11

RAWRITE3.COM 11
rcp 74
rdev 74
readmsg 74
reboot 74
ref 74
renice 74
rev 74
rlogin 74
rlogind 74
rm 74
rmail 74
rmdir 74
rootflags 74
route 74
rsh 74
rshd 74
rsmtp 74
runlevel 74
runq 74

S

script 75
sdiff 75
sed 75
selection 44, 75
sendmail 75
setfdprm 75
setfont 75
setleds 75
setmetamode 75
setserial 75
setsid 75
setterm 75
sh 75
showfont 75
showkey 75
shutdown 49, 75
size 75
sleep 75
smail 10, 65, 75
smtpd 75
sort 75
split 75
strings 75
strip 75
stty 75
su 75
sum 75
swapdev 75
swapoff 75
swapon 75
sync 75
syslogd 75

T

tac 75
tail 75
talk 75
tar 75
tbl 75
tee 75
telinit 75
telnet 75
telnetd 75
test 75
tfmtodit 75
time 75
tload 75
top 75
touch 75
tr 75
true 76
tsort 76
tty 76
tune2fs 15, 76

U

ul 76
umount 76
uname 76
uncompress 76
unexpand 76
uniq 76
unprotoize 76
update 76
updatedb 76
uptime 76
users 76
utmpdump 76
uudecode 76
uuencode 76
uugetty 76
uupath 76
uuwho 76

V

vdir 76
vgaset 47
vi 67, 76
vidmode 76
virec 76
vlock 76
vmlinuz 1, 76

W

w 76
wall 76
wc 76
wdsetup 42, 76
whatis 76
whereis 76
who 76
whoami 76
wnewmail 76
write 76

X

xargs 76
xdm 48

Y

yes 76

Z

zcat 76
zcmp 76
zdiff 76
zdump 76
zforce 76
zgrep 76
zic 76
zmore 76
znew 76

Concept Index

3

3c503 42

A

activating swap partitions 26
active flag 57
adding users 44
anonymous ftp 7

B

base disk 9
Base package 28
block size 51
boot disk 9
boot floppy
  conventional 37
  using LILO 35
boot sector 51
bootable partitions 57
booting
  from MCC boot disk 20
  with read-only / 21
BSD copyright 5
bugs 5

C

changing size of partitions 51
checksum 29
configuration files 44
configuring TCP/IP 42
console, virtual 2
contents of MCC distribution 3
conventional boot floppy 37
coprocessor emulation 2
Copyright 4
creating a file system 26
cylinders 50

D

DE-600 42
Debian version 3
demand loading 2
density 51
disaster recovery
  using MCC boot floppy 64
disk space required
  for MCC Interim Linux 16
  for MCC packages 15
  for MCC source files 16
  for swapping 17
  for TEX 16
  for X386 16
disks 1, 50
double density floppy 51

E

Ethernet cards 1, 42
ext file system 2, 27
ext2 file system 27
extended partitions 51

F

features of Linux 2
file system
  creating 26
  ext 2, 27
  ext2 27
  minix 27
  msdos 2, 60
  partitions 16
  proc 2
  repairing 27
file system standard 66
  xenix 2
floppy disk installation 11
force LILO to prompt 63

G

geometry of a disk 50
getting MCC Linux
  by anonymous ftp 9
  from the MCC shop 13
GNU copyleft 4
GNU library license 5
graphics 1

H

hackers 49
hardware 1
heads 50
hex code for root 20
high density floppy 51
HP-LAN 42

I

initialising swap partitions 25
installing 28
installing LILO 35
installing MCC packages
  from a DOS directory 11, 33
  from a Linux directory 33
  from the root floppy 28
  using floppy disks 11, 33
  using NFS 13, 33
interim 4

K

kernel parameters 45
  recompiling 41
kernels, precompiled 9
keyboard map files 30
keyboard maps 45

L

libraries, shared 2
LILO 35
  documentation 65
  forcing boot prompt 63
  removing from hard disk 37
Linux 1
ıLinux Installation and Getting Started ... 67
ıLinux System Administrator's Guide 67
logical partitions 51

M

manual pages, unformatted 10
map files, keyboard 30
maps, keyboard 45
maths coprocessor 2
maximum number of partitions 52
MCA bus architecture 1
MCC 3
MCC shop 13
memory 1
memory, shared 2
memory, virtual 2
minix file system 27
  30 character limit 27
MIT copyright 5
MIT X Consortium 5
mounting partitions 28
mouse 44
msdos file system 2, 60
`mtools` 60
multi-tasking 2

N

`NE2000` 42
network installation 13
new users 44
NFS options 5

P

paging 17
partition ID codes 58
partition tables 51
partitions 51
  editing with fdisk 23, 50
  file system 16
  hex code at boot 20
  swap 17
planning partitions 16, 17
platters 50
precompiled kernels 9
preparing a boot floppy
  from DOS 11
  from Linux 11
preparing to install Linux 15
primary boot block 51
primary partitions 51
proc file system 2
protected mode 2
PS/2 is unsupported 1
public domain software 5

R

ramdisk as root 3
recompiling the kernel 41
recovering from disasters 63
repairing a file system 27
resizing DOS partitions 51
root disk 9
root user 49

S

SCSI controller 1
sectors 50
selection 44
shared libraries 2
shared memory 2
shutting Linux down 49
single user mode 63
source files 3
sparse files 29
standard, Linux file system 66
superuser 49
swap files 17
swap partitions 17

activating 26
initialising 25
maximum size 26
swap space 17
system configuration files 44

T

TCP/IP
configuring 42
TEX 16
time zone data 10
tracks 50

U

unformatted manual pages 10
Unix 1
upgrading an existing system 4
upgrading MCC Interim Linux 64
users, adding 44

V

verifying files 29
virtual console 2
virtual memory 2

W

warnings 5
WD8003 42
WD8013 42

X

X386 16
xenix file system 2

Table of Contents

1. Introduction 1

1.1. What is Linux? 1
1.2. What are the MCC Interim versions of Linux? 3
1.3. Copyright and conditions of distribution 4
1.4. Bugs and warnings 5

2. Getting a Copy of MCC Interim Linux 7

2.1. Getting MCC Interim Linux by anonymous ftp 7
2.2. Getting MCC Interim Linux from the MCC shop 13
2.3. Installing MCC Interim packages using NFS 13

3. Preparing to Install Linux 15

3.1. Disk space required for Linux files 15
3.2. Swap space required for Linux 17

4. Installing or Updating MCC Interim Linux 19

4.1. Overview of an MCC installation or upgrade 19
4.2. Booting from the MCC boot disk 20
4.3. Editing the partition tables 23
4.4. Preparing swap and file system partitions 25
4.4.1. Initialising a swap partition 25
4.4.2. Activating a swap partition 26
4.4.3. Creating a file system 26
4.4.4. Checking and repairing an existing file system 27
4.5. Installing basic files from the MCC disks 28
4.5.1. Installing base packages using the '`ro`' option 28
4.5.2. Installing base packages using the '`ramdisk`' option 31
4.6. Installing the MCC Interim packages 32
4.7. Creating a boot floppy 35
4.7.1. Installing LILO on a floppy or hard disk 35
4.7.2. Making a conventional boot floppy 37
4.7.3. Making a special boot floppy using LILO 38

5. Tailoring MCC Interim Linux to Your Taste 41

5.1. Recompiling the kernel 41
5.2. Setting up Ethernet cards 42
5.3. Configuring the *tcpip* package 42
5.4. Editing the system configuration files 44
5.5. Passing parameters to the kernel 45
5.6. Installing the XFree86 software 47

6. Using Linux 49

6.1. Caring for your Linux system 49
6.2. `fdisk`: the Linux partition table editor 50

6.2.1. Disks and how they are described 50
6.2.2. Dividing up your disk 51
6.2.3. The `fdisk` command 53
6.2.4. Deleting and adding partitions 55
6.2.5. Active flags and system types 57
6.2.6. Extra commands for experts 58
6.2.7. Warnings for `fdisk` users 59
6.3. The *msdos* file system 60
6.4. Special commands for DOS floppies and partitions 60
6.5. Commands peculiar to Linux 63
6.6. Recovering from Fatal Disasters 63

7. Available Documentation for Linux 65

7.1. Documentation available on line 65
7.2. Collections of Frequently Asked Questions 65
7.3. The Linux HOWTO documents 66
7.4. The Linux file system standard 66
7.5. Other information about Linux 67

Appendix A. Acknowledgments 69

File Index 77

Program Index 79

Concept Index 85