Linuxita: the Minimal Working Set?

Peter T. Breuer

Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, ETSI Telecomunicación, Ciudad Universitaria, E–28040 Madrid, Spain.

Abstract

How small can a useful Linux system get? Linuxita fits on 4.7M of disk apace, works well in 3M of RAM on a 386sx, and the compressed tar file fits on a single 1.44M floppy. Is there any reason to be running DOS on legacy systems?

Introduction

When my carefully manicured Linux system had to be sent away for a long-postponed service call, I turned to an abandoned 386 for rescue. *Baby* allows me to keep my office hours as irregular as I like them to be, with email and the pcmcia modem, and I had been desperate to find a stopgap. I had prised a replacement for the 386's failed power supply from a backyard factory via long-distance, and lo! – the hard disk turned, dusty or no.

After much fuss in the dead month of August, I had taken delivery of a new external modem from the states and picked up a mains voltage converter to run it off – non-trivial shopping in those weeks. A few days before zero hour, I booted DOS on the old machine and it upped and went as though it had not been down for a year and a half. I called up Windows 3.1, played a little, and saw that it would not do. I had forgotten all about Windows on a 16MHz 386sx! Linux had to be installed on the 386 or I would go crazy while *baby* was gone. The following is the story of how to fit a Linux system into the minimum of disk space and RAM, and be happy.

Gosh, Darn, Preparations

I thought it might be a problem to fit Linux into the 20M disk of the 386, but knew it was room enough; a pared-down system plus a minimal X goes 15M at most. The problem would lie in separating out just the right pieces and getting the installation order so that nothing overflowed the available space at any stage. I would have to install piecemeal from the running system on *baby* over the countdown period.

I was too optimistic. I had problems backing up *baby*. The RAM on my office machine is faulty and I cannot copy over large files without errors. To cut the story short, I ran the office machine in DOS and made the transfers at 115K baud through the parallel port with

FastWire on both it and *baby*, the latter under dosemu. In the end, I never got time to install Linux on the 386 before *baby* left. I was lucky to finish the backup.

Then more problems. I had trouble setting up communications with the 386. There was no space for FastWire in the cramped 20M and no time to relearn my old setup in order to see how I could make room. Worse, everything under Windows on the old machine seemed to freeze the system, for the unknown reasons that normally apply. Its only serial port applications – apart from an ancient copy of kermit – were Windows applications!

In the end I found a small but modern serial package for DOS on the Garbo ftp archive site (garbo.uwasa.fi). I tried three or four packages before finding one (QVT) that both worked and fitted in the disk! To download it, I used DOS kermit to set up a 9600 baud direct login to the office machine through the serial port, then uuencoded and cat'ed the files past the screen while logging the session (I had a working DOS uudecode). That gave me an application that could set up a serial port connection at 38.4K under DOS, and I had an external 14.4K modem on standby. So when the courier arrived an hour early with the customs forms for *baby* not yet typed I felt the situation was livable.

But was 386sx hardware really compatible with a modern kernel? That was an old system. Was 3M of RAM enough? The HOWTO's said yes, but when did they date from? And because I could not afford to be without a system at home, I would have to keep DOS working on the 386 while I tried putting Linux either on top of it or in whatever space I could scrounge on the disk. I had not foreseen that.

More, I needed to keep some DOS installations like the C compiler and LaTeX alive in case Linux installation failed. I did not have any more room (or time!) to back those up to the office machine too. And most of the 20M disk was occupied by a Stacker compressed file system so I only had about 2M accessible via the bare FAT. I would have to fit the initial installation into something like 2M of disk space!

Revised Estimates

I took a close look at the office system. I keep that at a fairly standard Slackware 2.1 a.out installation, plus newer kernels and anything else I find I need, but I have not kept disk space down. How little could I really cut it down to? I needed /bin, /sbin, a kernel, the configuration files in /etc, the dynamic libraries in /lib, and not much else, as far as I could guess. To my astonishment, du said that was not very much at all! Six or seven megabytes. And I could probably trim it some.

Over the next few days I familiarized myself with the old 386 and DOS. I stripped out utilities I felt I could lose. I removed the permament Windows swap file and shrank the Stacker file system down. I collected fips and presizer for DOS and checked out the partition table. I found a disk defragmenter on one of my old archive diskettes. I no longer had fdisk on the 386 but I copied it from the office system and set up setver to cover up the difference in DOS versions (and languages!). I relearned my elaborate multi-boot setup on the 386 – which dynamic driver loader worked with which driver and which memory manager, which I needed right then and which I might need later. In the end I found myself with precisely

4.7M of free disk space. After a couple of fumbles, I had `fips` set up a new 4.7M partition at the end of the 20M disk. The method – if not the dimension! – is familiar to anyone who has installed Linux while preserving an existing DOS partition.

I had been experimenting on the office system too. In Linux mode, her name is *monica*, and she runs as part of a campus-wide net. *Monica* has one small 16M partition that I do not use for much of anything and which I have been meaning to set up as a spare root file system. What with the RAM fault, I can never be sure when *monica* will throw a fit that eats the file system instead of crashing more-or-less gently (she has since done the inevitable, and I am proud to say that I managed to put most of the 100M of nameless `lost+found` files back in the right places in an afternoon), and I would feel much safer with a separate root. I cleared out the partition and filled it with the files I thought I would need on root in the 386, and nothing else.

I will say that the documentation did not help me learn how to boot from a different partition with Lilo. It is easy when you know the trick – you really do have to just change the root partition name in `lilo.conf` and run Lilo – but it took me a day's experimenting. Along the way, I discovered that one can set up Lilo to boot from a kernel on a different partition too, even a DOS partition. Just change the location for the image in `/etc/lilo.conf`. The trick is to give the image location relative to the current root, not relative to the intended root. That is not as obvious as it sounds.

I found that almost nothing in the Slackware 2.1 `/bin` directory can be done without if the `init` boot sequence is to work. The `/etc/rc.d/` files are a flexible, well-designed and integrated system, but they exercise, or can exercise, quite a few unexpected utilities. I noticed that the Slackware root (and boot) diskette contains a much simplified `rc.d` system and thought about borrowing it. But I like and understand the `rc.d` system as it is, so I decided to hang on to it and live with the overhead. I could always change my mind later. But `/bin` came in at just under 1.2M, including both `bash` and `tcsh` shells (obviously, I only needed the first of these for the `init` sequence, and could even have got away with a lighter shell if I were only thinking about the `rd.d` sequence, but I could not live with a system on which I could not run decent shell scripts) so I would just have to be prepared to rethink later. One surprise was that I needed `test` from `/usr/bin`. It is needed in `rc.[SM0]`, at least. I had thought that a Unix system was meant to be able to boot up without using anything from `/usr`! (`sleep` is also in the wrong place in the current ELF slackware distribution).

Pretty soon, I had a bootable partition on *monica* with about 8M of files in it. I had not been cutting files out so much as throwing files in wholesale when I detected a need, so, encouraged, I went back to my 386, used `fdisk` again to split out a 1M partition from the 4.7M for use as swap and started to think about the important things ahead. What would I call the new system! Thanks to some unjustifiable extra spending some years ago, it has 3M of RAM to complement the 16MHz processor. One day I will be able to afford a coprocessor. *Bambam* was clearly an appropriate name for it and a choice that I look back on with satisfaction. 3.7M of disk space, 1M of swap and 3M of RAM would make for a mighty machine.

Installation

Bambam and I went back home and I used DOS and the modem to start transfering pieces that I reckoned I needed out of *monica*'s trial setup. I used a floppy as an intermediate storage area. I had no room to do anything else. Catch number one turned out to be that I could not use the ramdisk on the Slackware bootdisk kernel (I do not have enough memory on *bambam*). I had to use the Slackware rootdisk as my root device. With only one floppy slot, that left me stuck. I needed the floppy both as a root filesystem and as an i/o device. So I sighed, scrapped some more favourite utilities from my DOS partition, then squeezed the tar files onto it and tried again. Now I know that I should have made my own bootdisk using the minimally configured kernel on *monica*. I have had many communications since with people seeking to set up Linux systems in 3M of RAM or less, and in most cases the mistake they have been making has been using the slackware boot kernels. Those will take up at least 1.5M of RAM with all the compiled-in drivers, often leaving too little to boot `init` and the standard `getty`'s and daemons, plus a shell. *Monica*'s kernel takes up 1068K (and loads extra kernel modules when needed). Looking back, it is amusing that I made this mistake too.

System binaries

Monica's little trial partition had given me a guide as to what I really needed and what would be nice, but could be done without. So this time round I was careful to include the minimum. I have already mentioned that I needed about all of slackware 2.1 `/bin`. The list is in Figures 1,2,3 and 4.

As remarked, I voted to retain `bash`, despite its size, or I could have no fun writing shell scripts. I was surprised, but I could not do without `cut`. The `domainname` script I kludged as a call to `hostname` with the `-d` option. It would also be possible to use `yp-domainname` under other circumstances.

I have no idea if I had scripts that needed `getoptprog`, but it did not seem worth worrying about. The `ipmask` program has similar status. *Monica* is hooked up to the campus network the whole time, and I am sure that she needs it, but *bambam* could probably do without. Note that I put `less` here, and replaced `more` with a symlink to `less`.

The `ps` utility is rather costly – it also means compiling the `proc` filesystem into the kernel – but I hate to be without it. Likewise for `setterm`, though with less verve. Note that I included `sln` here. It is a statically linked version of `ln` that I have learned can get one out of a hole when playing with dynamic libraries.

To comment on the last (`zls`) link – I am running the ZLIB versions of the dynamic libraries in order to make use of compressed data files wherever possible, so I need `zls` as a way of calling `ls` and avoiding the interpretation induced by on-the-fly decompression. For the rest – I needed `tar` then and felt that I might need it again. The `tcsh` is my normal shell and I had to have it. I might have replaced `touch` with a shell script, but in the end I did not.

```
total 1181
-rwxr-xr-x   1 root     bin          1248 Sep 17  1994 arch
-rwxr-xr-x   1 root     bin        295940 Sep  5  1994 bash
-rwxr-xr-x   1 root     bin          4840 Nov 25  1993 cat
-rwxr-xr-x   1 root     bin          9220 Jul 20  1994 chgrp
-rwxr-xr-x   1 root     bin         13316 Jul 20  1994 chmod
-rwxr-xr-x   1 root     bin         13316 Jul 20  1994 chown
lrwxrwxrwx   1 root     root           17 Sep  3 07:18 compress -> /usr/bin/compress
-rwxr-xr-x   1 root     bin         21508 Jul 20  1994 cp
lrwxrwxrwx   1 root     root            4 Sep  3 07:19 csh -> tcsh
-rwxr-xr-x   1 root     bin          5192 Nov 25  1993 cut
-rwxr-xr-x   1 root     bin         19872 Mar 23  1994 date
-rwxr-xr-x   1 root     bin         17412 Jul 20  1994 dd
-rwxr-xr-x   1 root     bin         13316 Jul 20  1994 df
-rwxr-xr-x   1 root     root         1848 Aug 28 01:38 dirname
-rwxr-xr-x   1 root     bin          1752 Sep 17  1994 dmesg
lrwxrwxrwx   1 root     root            8 Sep  3 07:18 dnsdomainname -> hostname
-rwxr-xr-x   1 root     root           26 Sep  6 06:04 domainname
-rwxr-xr-x   1 root     bin         13316 Jul 20  1994 du
```

Figure 1: The `/bin` directory, a–d.

The `/sbin` directory I pretty much copied wholesale over to *bambam* (see Figure 5). I eliminated some utilities that I could rely on never needing again, such as `mk2efs`. Indeed, I removed all the file system `mk`...utilities. But I kept `fdisk` because I like to check that my partitions are still there occasionally! I eliminated `hdparm` because it does not work on my old drive)or at least, the version that I had did not seem to). I threw out various Slackware setup utilities (notably `setup`).

On the other hand, I had to keep, or perhaps include as extras, if one wishes to look at it that way, utilities to insert and remove dynamic kernel modules (`insmod`, `rmmod`, `modprobe`, `dep‘mod`). These come in the `modules-1.2.8` package on most archive sites. I use the kernel daemon (same package) to take some of the strain away. It unloads modules from RAM when they are not used for 30 seconds (by default), which is valuable.

I run both the `msdos` and `umsdos` modules in order to access the DOS partition on the disk. The `umsdos` file system means that I also have to keep around the `umssync` (a.k.a. `umssetup`) utilities to keep Linux up to date with the state of the DOS partition. Every so often I tend to makes some changes from DOS and then Linux will not see the new files (or lose the old ones) until `umssync` is run. It is worth having on the disk.

As can be seen in Figure 6, the only modules I chose to have available on *bambam* cost just under 80K of disk space. There was no need to keep `binfmt_elf.o` around – I was not running ELF, nor any need for `dummy.o` – I was not running the net, but I thought I might possible play a little.

Note that I use the `bflushd` and `updated` daemons, instead of the larger daemon (which splits into two when run) that is on the standard distributions. As I recall, I got these from the

```
-rwxr-xr-x   1 root     bin          3312 Mar 23  1994 echo
-rwxr-xr-x   1 root     bin           326 Mar 23  1994 false
-rwxr-xr-x   1 root     bin          2456 Oct 17  1994 free
-rwxr-xr-x   1 root     bin          1912 Sep 17  1994 getoptprog
lrwxrwxrwx   1 root     root            4 Sep  3 07:18 gunzip -> gzip
-rwxr-xr-x   1 root     bin         46084 Sep  5  1993 gzip
-rwxr-xr-x   1 root     bin          4256 Nov 25  1993 head
-rwxr-xr-x   1 root     bin          3536 Sep 17  1994 hostname
-rwxr-xr-x   1 root     bin          2000 Aug 16  1994 ipmask
-rwxr-xr-x   1 root     bin          2028 Sep 17  1994 kill
-rwxr-xr-x   1 root     bin          4228 Oct 17  1994 killall
-rwxr-xr-x   1 root     root        54276 Sep  3 02:10 less
-rwxr-xr-x   1 root     bin         13316 Jul 20  1994 ln
-rwxr-xr-x   1 root     bin          6752 Sep 18  1994 login
-rwxr-xr-x   1 root     bin         25604 Feb 26  1994 ls
```

Figure 2: The `/bin` directory, e–l.

`apm-0.5` package. They are specially tuned for laptops and other small machines (the kernel I was using on *bambam* had APM compiled into it, although I doubt if *bambam* knew anything about it). I have not been able to compile the assembler with newer compiler versions, so I have been passing these on to myself as a binary inheritance for some time now.

System libraries

My extravagance so far had left me with 2.25M free from the 4.7M partition minus 1M swap. I went back and remade the standard `/lost+found` directory in order to save myself 8K. The `mk2efs` sets the directory size to 12K to start with, which has always seemed pessimistic to me. On `/lib` itself, however, I saved a little more.

To my surprise, I did not need anything more than the basic `libc` and `libm`. I had been worried that I might need `libvga` or other exotica for some of the standard utilities, but no. The `linuxaout-uncompress.o` contains the to-be-preloaded library functions for the ZLIB library modification. I prefer using the preloaded module rather than altering `libc`, which is the other method of getting ZLIB up and running. But it obviously uses more space. I need the `ld.so` version 1.7.3 or better to make dynamic preloading work. With LD_PRELOAD set to point to it, on each call to the dynamic libraries the preload module is scanned first. The module intercepts reads from compressed files and uncompresses them through a pipe. It can save considerable space.

I believe that I am supposed to hard-link `ld.so`, but a soft link seems to work fine.

The libraries consumed 780K. At this point I had 1.45M of partition space available and the configuration files in `/etc` were still to come.

```
-rwxr-xr-x   1 root     bin         13316 Jul 20  1994 mkdir
-rwxr-xr-x   1 root     bin          9220 Jul 20  1994 mkfifo
-rwxr-xr-x   1 root     bin          9220 Jul 20  1994 mknod
lrwxrwxrwx   1 root     root            4 Sep  3 12:24 more -> less
-rwxr-sr-x   1 root     bin         17424 Sep 17  1994 mount
-rwxr-xr-x   1 root     bin         13316 Jul 20  1994 mv
-rwxr-xr-x   1 root     bin         21508 Oct 17  1994 ps
-rwxr-xr-x   1 root     bin          1368 May  4  1994 pwd
-rwxr-xr-x   1 root     bin         13316 Jul 20  1994 rm
-rwxr-xr-x   1 root     bin          9220 Jul 20  1994 rmdir
-rwxr-xr-x   1 root     bin          7536 Sep 17  1994 setserial
-rwxr-xr-x   1 root     bin         10968 Sep 17  1994 setterm
lrwxrwxrwx   1 root     root            4 Sep  3 07:18 sh -> bash
-rwxr-xr-x   1 root     bin         11132 Sep 17  1994 sln
-rwxr-xr-x   1 root     bin         19356 Mar 23  1994 stty
-rwsr-sr-x   1 root     bin          5492 Mar 23  1994 su
-rwxr-xr-x   1 root     bin            64 Sep 17  1994 sync
```

Figure 3: The /bin directory, m–s.

```
-rwxr-xr-x   1 root     bin        140292 Jun 24  1994 tar
-rwxr-xr-x   1 root     bin        209924 Jul 11  1993 tcsh
-rwxr-xr-x   1 root     bin         21508 Jul 20  1994 touch
-rwxr-xr-x   1 root     bin           328 Mar 23  1994 true
-rwxr-xr-x   1 root     bin          8888 Sep 17  1994 umount
-rwxr-xr-x   1 root     bin          2772 Mar 23  1994 uname
lrwxrwxrwx   1 root     root            4 Sep  3 07:18 zcat -> gzip
lrwxrwxrwx   1 root     root            2 Sep  3 23:33 zls -> ls
```

Figure 4: The /bin directory, t–z.

System configuration

I edited down the number of `agetty`'s started in `/etc/inittab` to two. Any more was an extravagance from my point of view. Even if they did share code. Two virtual consoles is enough. Note that I preferred `agetty` to `getty_ps` or other alternatives for reasons of space. It is the simplest.

In the end, I had `/etc` down to just over 100K. The biggest files are `magic` (for the `file` command) and `termcap`. The latter can be edited down and the former I compressed. Using ZLIB means that it is read alright.

I left the `init` sequence files in `/etc/rc.d` as standard – for a cost of 27K. And out of nostalgia (or hope?) I let one or two network configuration files stand – `resolv.conf`, for example. I did optimize some things in `/etc`, however. I emptied the `skel` subdirectory (I was not going to make new users). I made sure that the `locale` subdirectory was empty, and checked that `fs` only contained soft links and no executables. The real executables should go in `/sbin`.

1.35M left and counting.

Application binaries

What applications would I need? My requirements were connectivity and local editing. The rest I could rely on my remote systems for.

Communications

Since I wanted to work a modem, I needed `minicom` as a communications package. It is small and neat (about 80K) and I like it. Perhaps there is something more suitable out there, but I do not know about it. It needs one or two configuration files in `/var/lib/minicom`, and nothing else. Each user also gets a local configuration file `~/.minirc` and a phone number list, but there would only be one user – apart from root. A few extra kilobytes. By this stage I was definitely counting kilobytes. A separate utility, `runscript`, is also required in order to parse the login scripts. 15K more. And I needed to upload and download, so I needed the zmodem protoocols. That requires `lrz` and `lsz` and some soft links. 60K more.

Editor

For each individual, there is only one choice of editor. For me, it is `vi` – rather `vim`, the multi-window version. Perhaps an `emacs` person would have to settle for `joe` as a poor substitute in the disk space available, but I did not have to compromise. That cost 185K, plus my configuration file.

Sundries

Certain applications are neither necessities nor luxuries. One can do without them, but life would be sadder. In that category I count the compression utilities `gzip` and `gzexe`, as well as the ever-entertaining `top`. At least it is useful for tuning. `tset` (and `reset`, only a shell script) are also too much bother to miss out. `loadkeys` I needed for my UK keyboard layout, but I trimmed `/usr/lib/kbd/keytables` down to just the map that I needed. See Figure 9 for the complete list that I left in `/usr/bin`. It occupied just under 600K in total, which left me with 750K.

Luxuries

Although my institute's modems are all even-parity, I had had success in getting `term` to work across the link (the 1.9 version, not the later series, for some reason), so I bundled it too. Multiplexed unix logins to a big system are a blessing not to be missed. I would have used `slirp`, but then I would have had to install some network utilities, and I did not have the space. What is more, I have never discovered how to get slirp to cope with an even-parity modem line. `term` goes in `/usr/local/bin` with the remote shell (`trsh`) and shutdown utility (`tshutdown`). For fun I added the termified uploader too. All these utilities are the same size and I suspect that they may be the same inside. Perhaps soft links might work instead of renamed copies? I did not experiment. They came to just under 100K, with the configuration file. 650K to go.

And although I can read my dos floppies by mounting `/dev/fd0` as an `msdos` type file system, I prefer to use the `mtools` suite. It only requires a single 42K executable, and all the variants (`mwrite`, `mread`, `mcopy`, etc.) are soft links to it. I put those in `/usr/local/bin` too. 600K left.

Application libraries

Fortunately, my choice of applications did not require any extra dynamic libraries.

Overheads

What about all the uncountable bits and pieces, like directories (which take up room) and a few scripts such as `MAKEDEV` that one should never risk being without? Those take up a bit of extra space. Then there is the compressed kernel image itself, and some log files and other administrivia. All in all, about another 3500K of disk space.

Of course I chose not to run `syslog` and friends (the messages just scroll pass the console instead) and ponted the log files at `/dev/null`. I did not need the Slackware `/var/log/packages` directory either, or `scripts`. I did not need most of the `/var/spool` subdirectories since I was not running `cron` or `at`, or a printer or mail. So I saved a little there.

There is a small penalty for using a system – instead of looking at it admiredly. I had to install a home directory for myself. Since I had so much space, I gave root its own home too, just as one is supposed to. In the end I had about 250K of usable space left free on the disk. Enough to edit files on, and pass files to and from over the modem – just as I had required.

Conclusion

It is indeed possible to run Linux on a 386sx machine with just a few megabytes of disk space available, and very limited RAM. With less RAM than the 3M I had I might have needed to use a non-production kernel, but the standard 1.2.13 kernel worked well for me, when cut down and mounting modules dynamically by need. I was able to shoehorn the file system into a little over 3.6M, and used a 1M swap partition. The performance is at least as good as with DOS, and I am a lot happier.

```
/sbin:
total 252
drwxr-xr-x   2 root     uucp         1024 Oct  4 04:28 ./
drwxr-xr-x  17 root     root         1024 Sep 10 21:26 ../
-rwxr-xr-x   1 root     bin          6504 Sep 17  1994 agetty*
-rwxr-x---   1 root     bin          5336 Apr 12  1994 badblocks*
-rwxr-xr-x   1 root     users         124 Apr  1  1995 bdflushd*
-rwxr-xr-x   1 root     bin          4568 Sep 17  1994 clock*
lrwxrwxrwx   1 root     root           14 Sep  3 06:49 depmod -> /sbin/modprobe*
-rwxr-x---   1 root     bin         55936 Apr 12  1994 e2fsck*
-rwxr-xr-x   1 root     bin         18136 Sep 17  1994 fdisk*
-rwxr-xr-x   1 root     bin          3232 Sep 17  1994 fsck*
-rwxr-xr-x   1 root     bin          9220 Nov 25  1993 halt*
-rwx--x--x   1 root     bin          1392 Jul 14  1993 hostname_notcp*
-rwxr-xr-x   1 root     bin         17412 Nov 25  1993 init*
-rwxr-sr-x   1 root     root        18080 Jun 11  1995 insmod*
-rwxr-xr-x   1 root     bin          1920 Sep 17  1994 kbdrate*
-rwx------   1 root     root         5336 Jun 11  1995 kerneld*
lrwxrwxrwx   1 root     root            6 Sep  3 06:49 ksyms -> insmod*
-rwxr-xr-x   1 root     root          118 Jun 11  1995 lsmod*
-rwxr-xr-x   1 root     bin          5212 Sep 17  1994 mkfs*
-rwxr-x---   1 root     bin          4000 Nov 30  1993 mksuper*
-rwxr-xr-x   1 root     bin          2452 Sep 17  1994 mkswap*
-rwxr-xr-x   1 root     bin         18816 Jun 11  1995 modprobe*
lrwxrwxrwx   1 root     root           10 Sep  3 06:49 mount -> /bin/mount*
lrwxrwxrwx   1 root     root            4 Sep  3 06:49 ramsize -> rdev*
-rwxr-xr-x   1 root     bin          3852 Sep 17  1994 rdev*
lrwxrwxrwx   1 root     root            4 Sep  3 06:49 reboot -> halt*
lrwxrwxrwx   1 root     root            6 Sep  3 06:49 rmmod -> insmod*
-rwxr-x---   1 root     bin          9220 Nov 27  1993 rmt*
lrwxrwxrwx   1 root     root            4 Sep  3 06:49 rootflags -> rdev*
-rwxr-xr-x   1 root     bin         13316 Nov 25  1993 shutdown*
lrwxrwxrwx   1 root     root            4 Sep  3 06:49 swapdev -> rdev*
lrwxrwxrwx   1 root     root            6 Sep  3 06:49 swapoff -> swapon*
-rwxr-xr-x   1 root     bin          2428 Sep 17  1994 swapon*
lrwxrwxrwx   1 root     root            4 Sep  3 06:49 telinit -> init*
-rwxr-x---   1 root     bin          1936 Nov 30  1993 testfs*
lrwxrwxrwx   1 root     root            7 Sep  3 06:49 udosctl -> umssync*
lrwxrwxrwx   1 root     root           11 Sep  3 06:49 umount -> /bin/umount*
lrwxrwxrwx   1 root     root            7 Sep  3 06:49 umssetup -> umssync*
-rwxr-xr-x   1 bin      bin         23832 Jul 18  1995 umssync*
-rwxr-xr-x   1 root     users         160 Apr  1  1995 updated*
lrwxrwxrwx   1 root     root            4 Sep  3 06:49 vidmode -> rdev*
```

Figure 5: The /sbin directory.

```
/lib/modules/1.2.13/fs:
total 63
drwxr-xr-x   2 root     uucp         1024 Sep  3 09:54 ./
drwxr-xr-x   5 root     uucp         1024 Jul  9  1995 ../
-rw-r--r--   1 root     root        30083 Jul  9  1995 msdos.o
-rw-r--r--   1 root     root        27307 Jul  9  1995 umsdos.o

/lib/modules/1.2.13/misc:
total 9
drwxr-xr-x   2 root     uucp         1024 Aug 27 19:32 ./
drwxr-xr-x   5 root     uucp         1024 Jul  9  1995 ../
-rw-r--r--   1 root     root         7253 Jul  9  1995 binfmt_elf.o

/lib/modules/1.2.13/net:
total 4
drwxr-xr-x   2 root     uucp         1024 Sep  3 09:43 ./
drwxr-xr-x   5 root     uucp         1024 Jul  9  1995 ../
-rw-r--r--   1 root     root         1367 Jul  9  1995 dummy.o
```

Figure 6: The `/lib/modules` directories.

```
total 779
lrwxrwxrwx   1 root     root           12 Sep  3 07:53 cpp -> /usr/bin/cpp
lrwxrwxrwx   1 root     root           11 Sep  3 07:53 ld.so -> ld.so.1.7.3
-rwxr-xr-x   1 root     root        20484 Jun 29 23:02 ld.so.1.7.3
lrwxrwxrwx   1 root     root           13 Sep  3 07:53 libc.so.4 -> libc.so.4.7.2
-rwxr-xr-x   1 root     root       634880 Apr 29 14:58 libc.so.4.7.2
lrwxrwxrwx   1 root     root           14 Sep  3 07:53 libm.so.4 -> libm.so.4.6.27
-rwxr-xr-x   1 root     root       110592 Feb 18  1995 libm.so.4.6.27
-rwxr-xr-x   1 root     root        22349 Jun 29 06:50 linuxaout-uncompress.o
drwxr-xr-x   9 root     uucp         1024 Sep  3 10:21 modules
```

Figure 7: The `/lib` directory.

```
/etc:
total 107
drwxr-xr-x   6 root     uucp         1024 Jan 25 23:48 ./
drwxr-xr-x  17 root     root         1024 Sep 10 21:26 ../
-rw-r--r--   1 root     root         1952 Jun 23  1995 DIR_COLORS
-rw-r--r--   1 root     root           11 Sep  6 00:07 DOMAINNAME
-rw-r--r--   1 root     root           19 Sep  3 08:38 HOSTNAME
-rw-r--r--   1 root     root           30 Sep  6 00:00 KEYTABLE
lrwxrwxrwx   1 root     root           14 Sep  3 06:46 X11 -> /var/X11R6/lib
-rw-r--r--   1 root     root         1324 Aug 26 20:31 conf.modules
-rw-r--r--   1 root     root          139 Sep  9 10:17 csh.cshrc
-rw-r--r--   1 root     root          787 Sep  9 10:16 csh.login
-rw-r--r--   1 root     root          443 Jan 24  1994 disktab
-rw-r--r--   1 root     root            0 Dec 24  1994 fastboot
-rw-r--r--   1 root     root         1182 Dec 13  1992 fdprm
drwxr-xr-x   2 root     uucp         1024 Jan  5  1995 fs/
-rw-r--r--   1 root     root          379 Sep  5 01:39 fstab
-rw-r--r--   1 root     root          369 Oct  2 23:44 group
-rw-r--r--   1 root     root           26 Mar  4  1995 host.conf
-rw-r--r--   1 root     root            4 Jan 28  1995 hostid
-rw-r--r--   1 root     root          402 Sep  3 08:39 hosts
-rw-r--r--   1 root     root           23 Jul 11  1995 hosts.term
lrwxrwxrwx   1 root     root            1 Sep  3 06:46 inet -> ./
-rw-r--r--   1 root     root         2745 Dec 30 22:26 inittab
-rw-r--r--   1 root     root           27 Jan 25 23:32 issue
-rw-r--r--   1 root     root           99 Sep  4 23:41 ld.so.cache
-rw-r--r--   1 root     root           45 Mar 18  1994 ld.so.conf
-rw-r--r--   1 root     root          609 Sep  3 12:40 lilo.conf
drwxr-xr-x   3 root     root         1024 Sep  3 23:42 locale/
pr--r--r--   1 root     root        47874 Sep 27  1994 magic|
-rw-r--r--   1 root     root           23 Jan 25 23:32 motd
-rw-r--r--   1 root     bin           123 Jan 25 23:48 mtab
-rw-r--r--   1 root     root          835 Sep  9 23:35 mtools
-rw-r--r--   1 root     root          233 Jan 28  1995 networks
-rw-r--r--   1 root     root          847 Sep  9 10:44 passwd
-rw-r--r--   1 root     root         1280 Sep  4 07:32 profile
drwxr-xr-x   2 root     uucp         1024 Sep 10 23:41 rc.d/
-rw-r--r--   1 root     root           39 Jan 28  1995 resolv.conf
-rw-r--r--   1 root     root           86 Jan 28  1994 securetty
-rw-r--r--   1 root     root           37 Jan  5  1995 shells
drwxr-xr-x   3 root     uucp         1024 Jan  5  1995 skel/
-rw-r--r--   1 root     root        24318 Jul  9  1995 termcap
-rw-r--r--   1 root     root          138 Jan 20  1995 ttys
lrwxrwxrwx   1 root     root           13 Sep  3 06:46 utmp -> /var/log/utmp
lrwxrwxrwx   1 root     root           13 Sep  3 06:46 wtmp -> /var/log/wtmp
```

Figure 8: The `/etc` directory.

```
/usr/bin:
total 575
drwxr-xr-x   2 root     root         1024 Sep 26 09:58 ./
drwxr-xr-x   9 root     root         1024 Sep 12 00:27 ../
lrwxrwxrwx   1 root     root            4 Sep  3 20:52 [ -> test*
-rwx--x--x   1 ptb      users        1976 Sep 11 23:55 basename*
-rwxr-xr-x   1 root     root           25 Sep  4 06:27 clear*
-rwxr-xr-x   1 501      users       13084 Sep  3 21:20 file*
-rwxr-xr-x   1 root     root        62468 Sep  3 00:03 grep*
-rwx--x--x   1 ptb      users        3823 Sep 11 23:47 gzexe*
-rwxr-xr-x   1 501      users        2648 Sep  3 18:33 ldd*
-rwxr-xr-x   1 ptb      users       37892 Sep  5 21:06 loadkeys*
-rwxr-xr-x   1 root     bin         25604 Jul 17  1994 lrz*
-rwxr-xr-x   1 root     bin         29700 Jul 17  1994 lsz*
-rwxr-sr-x   1 root     uucp        78852 Sep  2 23:55 minicom*
lrwxrwxrwx   1 root     root            3 Sep  3 12:33 rb -> lrz*
-rwxr-xr-x   1 ptb      users         121 Sep  5 21:05 reset*
-rwxr-xr-x   1 root     root        13316 Sep  9 01:20 runscript*
lrwxrwxrwx   1 root     root            3 Sep  3 12:33 rx -> lrz*
lrwxrwxrwx   1 root     root            3 Sep  3 12:33 rz -> lrz*
lrwxrwxrwx   1 root     root            3 Sep  3 12:34 sb -> lsz*
-rw-r--r--   1 root     root            0 Sep 26 09:58 screen.dump
-rwx--x--x   1 ptb      users       54276 Sep 12 00:31 sed*
lrwxrwxrwx   1 root     root            3 Sep  3 12:34 sx -> lsz*
lrwxrwxrwx   1 root     root            3 Sep  3 12:33 sz -> lsz*
-rwxr-xr-x   1 ptb      users        7672 Sep  5 21:05 tail*
-rwxr-xr-x   1 501      users       12228 Sep  3 18:36 test*
-rwxr-xr-x   1 501      users       21508 Sep 12 00:37 top*
-rwxr-xr-x   1 501      users       13316 Sep  4 04:20 tset*
lrwxrwxrwx   1 root     root            3 Sep  3 09:56 vi -> vim*
-rwxr-xr-x   1 root     root       185348 Sep  2 23:55 vim*
```

Figure 9: The `/usr/bin` directory.

```
/usr/local/bin:
total 144
drwxr-xr-x   2 root     root         1024 Sep 10 23:53 ./
drwxr-xr-x   5 root     root         1024 Sep 10 19:46 ../
lrwxrwxrwx   1 root     root            6 Sep  4 06:31 mcd -> mtools*
lrwxrwxrwx   1 root     root            6 Sep  3 13:02 mcopy -> mtools*
lrwxrwxrwx   1 root     root            6 Sep  3 13:02 mdel -> mtools*
lrwxrwxrwx   1 root     root            6 Sep  3 13:02 mdir -> mtools*
lrwxrwxrwx   1 root     root            6 Sep  3 13:02 mformat -> mtools*
lrwxrwxrwx   1 root     root            6 Sep 22 09:46 mlabel -> mtools*
lrwxrwxrwx   1 root     root            6 Sep  4 06:32 mmd -> mtools*
lrwxrwxrwx   1 root     root            6 Sep  4 06:35 mrd -> mtools*
lrwxrwxrwx   1 root     root            6 Sep  3 13:02 mread -> mtools*
-rwxr-xr-x   1 501      users       41988 Sep 12 00:37 mtools*
lrwxrwxrwx   1 root     root            6 Sep  3 13:02 mwrite -> mtools*
-rwxr-xr-x   1 root     term        49152 Sep  8 21:30 term*
-rwxr-sr-x   1 root     term        16384 Sep  8 21:31 trsh*
-rwxr-sr-x   1 root     term        16384 Sep  8 21:31 tshutdown*
-rwxr-sr-x   1 root     term        16384 Sep  8 21:31 tupload*
```

Figure 10: The `/usr/local/bin` directory.