Creating and Using Linux Emergency Recovery Disks

BETTER SAFE THAN SORRY

MARTIN MILNER

What would you do if the Linux system you spent many hours building suddenly wouldn't load? – due to a mistake during configuration? Re-install? What about your precious data? In this article, we'll explain the steps necessary to create a complete Linux system which will boot from floppy disks and allow you to perform essential recovery work like restoring a backup of your root filesystem. (You have got one, haven't you?)

So your Linux system is broken. Maybe you had problems with the hard disk or a power cut and then the fsck (filesystem check) of the root filesystem came up with loads of errors.

If you're used to using Windows 9x, you'll probably know about the Windows emergency boot disk you can create, but it doesn't allow you to do a great deal and it certainly won't load and run Windows. However, a basic Linux system can run off one or more floppy disks – yet still provide a basic set of essential tools.

If you bought an official Linux distribution from one of the main suppliers you may have received a recovery disk with it. Lucky you. If however, like many people you built a system off a magazine CD or similar, then you most certainly won't have one.

The disk set described here consists of a boot disk, a disk containing a root filesystem with a small set of tools and a utility disk to hold a number of additional utilities. The article assumes you have ramdisk support enabled in your kernel. If you haven't, then you will need to enable it.

Making a boot disk

The first disk we need to create is the boot disk. This contains a Linux kernel and the kernel loader LILO. It is possible to create a boot disk which also contains a root filesystem, (a 'boot/root' disk), but because of the small size of even HD floppy disks, the resulting system will be severely lacking in essential utilities.

By far the easiest way of creating a boot disk is by using the command mkbootdisk (see figure 1) like this:

```
mkbootdisk –verbose kernelversion (eg:- 2.2.16)
```

This command creates a stand-alone boot floppy for your running system. The most important parameter is the last one, which is the kernel version. Note that there are (at least) two versions of mkbootdisk, one which doesn't add the rescue option to */etc/lilo.conf*. Whichever version you've got after it finishes, mount the disk and edit the *lilo.conf* file until it looks similar to that in figure 2 and then rerun LILO like this:

```
mount -t ext2 /dev/fd0 /mnt/floppy (/mnt/➥
floppy must already exist)
vi /mnt/floppy/etc/lilo.conf
/sbin/lilo -v -r /mnt/floppy
umount /mnt/floppy
```

The 'ramdisk' option in *lilo.conf* ensures the ramdisk is big enough for the root filesystem we'll be

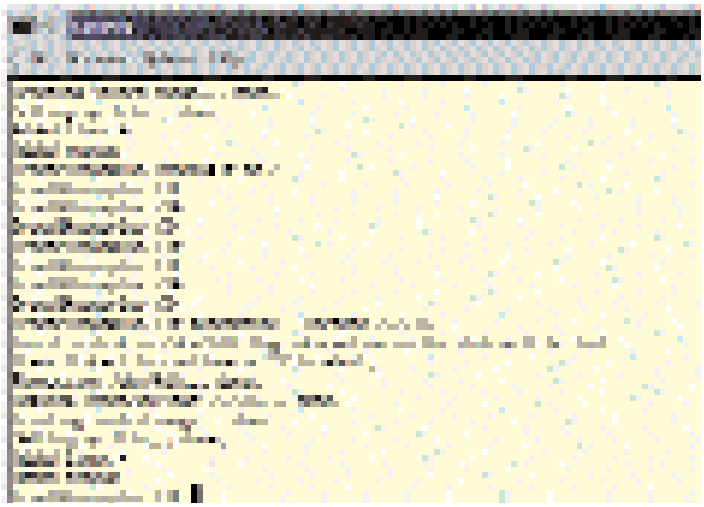

Figure 1 - Using mkbootdisk to make the boot floppy

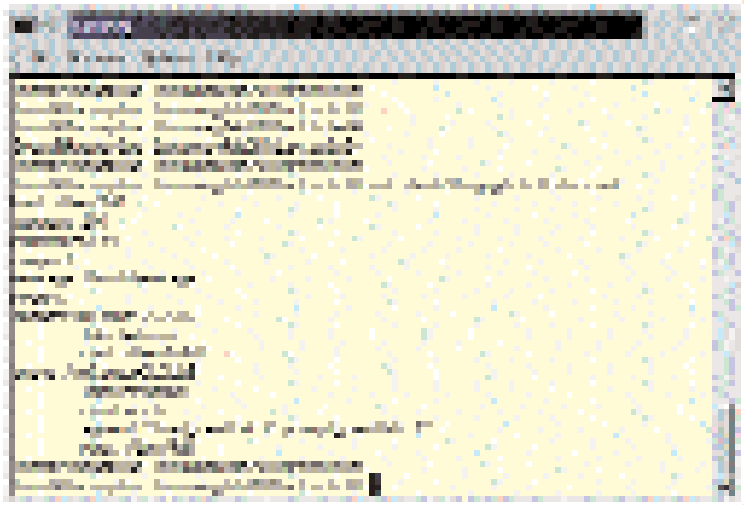

Figure 2 - How /etc/lilo.conf should look

creating below. The *compact* option speeds up the loading process and the *append* line tells the kernel to prompt for a root filesystem and load it into the ramdisk.

Once finished, you will have a floppy disk containing your current kernel, LILO and a number of other system files (see figure 3). When you reboot your machine with this disk inserted, LILO will give you the choice of booting up off your hard disk or typing in *rescue* to boot from floppy. After choosing *rescue*, you will eventually be asked for a disk containing a root filesystem, which is what we'll create next.

Creating a root filesystem

The root filesystem must contain everything needed to support a full Linux system. In other words:

1. The basic filesystem structure.
2. A minimum set of directories. (*/dev, /proc, /bin, /etc, /lib, /usr, /tmp,* etc.)
3. A basic set of utilities. (bash, ls, cp, mv, etc.)
4. A minimum set of config files. (*inittab, fstab,* etc.)
5. Devices. (*/dev/hd*, /dev/tty*, /dev/fd0,* etc.)
6. Runtime libraries to provide basic functions used by utilities.

To allow us to have as many files, utilities, etc. as possible in our root filesystem, we'll build a compressed filesystem. Obviously, this means we'll have to build it elsewhere. There are a number of ways of doing this.

1. Use a ramdisk. (*/dev/ramdisk or /dev/ram0*).
2. Use an unused hard disk partition.
3. Use a loopback device, which allows a disk file to be treated as a device. (For which you need specially modified mount and unmount commands.)

For this excercise, we'll assume you haven't got an unused partition or the disk space to create one and use a ramdisk. First, prepare the ramdisk:

```
dd if=/dev/zero of=/dev/ramdisk bs=1k count=➥
4000 (approx. 4Mb.)
```

Next, create the filesystem:

```
mke2fs -m 0 -i 2000 /dev/ramdisk
```

mke2fs will automatically detect the space available. The -i 2000 is to increase the amount of inodes to make sure we don't run out. Now make an appropriately named mount point (if you haven't done so before) and mount the new filesystem:

```
mkdir /mnt/ramdisk
mount -t ext2 /dev/ramdisk /mnt/ramdisk
```

Copy over the appropriate device files from the */dev* directory like this:

```
mkdir /mnt/floppy/dev
cp -dpR /dev/hda? /mnt/ramdisk/dev
```

Repeat the above for all the devices you might need. Next create the other directories on the floppy and then copy all the other files into them. See the boxout for an example of the required files and directories. Be especially careful that symbolic links are preserved. (Many of the library files in */lib* are links.)

Config files and finishing off

Some of the *config* files will need changing to reflect their intended use. See figure 4 for the contents of the files that will require editing. When you've done all that and are reasonably happy that all is well, do the following:

```
umount /mnt/ramdisk
dd if=/dev/ramdisk of=rootfs bs=1k
gzip -v9 rootfs
```

When *gzip* is finished, *rootfs.gz* contains the compressed root filesystem. Make sure that *rootfs.gz* will fit on a floppy disk. If it's too big unzip it, remount the filesystem as before, delete some stuff out of it and try the above again.

Finally, it's time to write it to floppy disk,

```
dd if=rootfs.gz of=/dev/fd0 bs=1k
```

Example of contents of a floppy root filesystem

```
/mnt/ramdisk:
bin     dev     etc     lib     mnt     mnt2    proc    root    sbin    tmp     usr     var

/mnt/ramdisk/bin:
bash    cat     chmod   chown   cp      date    dd      df      echo    false   grep    hostname
id      ln      login   ls      mkdir   mknod   more    mount   mt      mv      ps
pwd     rm      rmdir   sh      stty    su      sync    touch   true    umount  uname

/mnt/ramdisk/dev:
cdrom   cdu31a  console fd0     hda1    hda2    hda3    hda4    hda5    hda6    hda7    hda8
hda9    hdb1    hdb2    hdb3    hdb4    hdb5    hdb6    hdb7    hdb8    hdb9    hdc     hdc1
hdc2    hdc3    hdc4    hdc5    hdc6    hdc7    hdc8    hdc9    hdd1    hdd2    hdd3    hdd4
hdd5    hdd6    hdd7    hdd8    hdd9    kmem    mem     null    ram     ram0    ramdisk sda1
sda2    sda3    sda4    sda5    sda6    sda7    sda8    sda9    sdb1    sdb2    sdb3    sdb4
sdb5    sdb6    sdb7    sdb8    sdb9    tty0    tty1    tty2    ttyS1   zero

/mnt/ramdisk/etc:
conf.modules    fstab   gettydefs group   inittab issue   ld.so.cache     motd
nsswitch.conf   pam.d   passwd  profile rc      shadow  shells  termcap ttys    utmp    wtmp

/mnt/ramdisk/etc/pam.d:     other

/mnt/ramdisk/lib:
ld-2.1.1.so             ld-linux.so.2           libc-2.1.1.so           libc.so.6
libcom_err.so.2         libcom_err.so.2.0       libcrypt-2.1.1.so       libcrypt.so.1
libdl-2.1.1.so          libdl.so.1              libdl.so.1.9.5          libdl.so.2
libext2fs.so.2          libext2fs.so.2.4        libnsl-2.1.1.so         libnsl.so.1
libnss_files-2.1.1.so   libnss_files.so.2       libpam.so               libpam.so.0
libpam.so.0.66          libpam_misc.a           libpam_misc.so          libpam_misc.so.0
libpam_misc.so.0.66     libproc.so.2.0.0        libpwdb.so              libpwdb.so.0
libpwdb.so.0.58         libtermcap.so.2         libtermcap.so.2.0.8     libutil-2.1.1.so
libutil.so.1            libuuid.so.1            libuuid.so.1.2

/mnt/ramdisk/lib/modules/2.2.12-10/block:       loop.o

/mnt/ramdisk/lib/modules/2.2.12-10/cdrom:       cdu31a.o

/mnt/ramdisk/lib/security:      pam_permit.so

/mnt/ramdisk/mnt:       cdrom   floppy

/mnt/ramdisk/sbin:
depmod  fdisk   halt    head    init    insmod  kerneld lsmod   mingetty
mkswap  modprobe rmmod  shutdown sulogin swapoff swapon  tail    update

/mnt/ramdisk/var: log   run     tmp

/mnt/ramdisk/var/log:   wtmp
/mnt/ramdisk/var/run:   utmp
/mnt/ramdisk/var/tmp:   tmp
```

Example contents of a Utility disk

```
mnt/floppy:
bin     lib     lost+found      man     sbin    share
/mnt/floppy/bin:
cut     diff    du      find    gunzip  gzip    passwd  tar     vi
/mnt/floppy/sbin: chroot  fuser   lilo    mke2fs  mkfs    mkfs.ext2
```

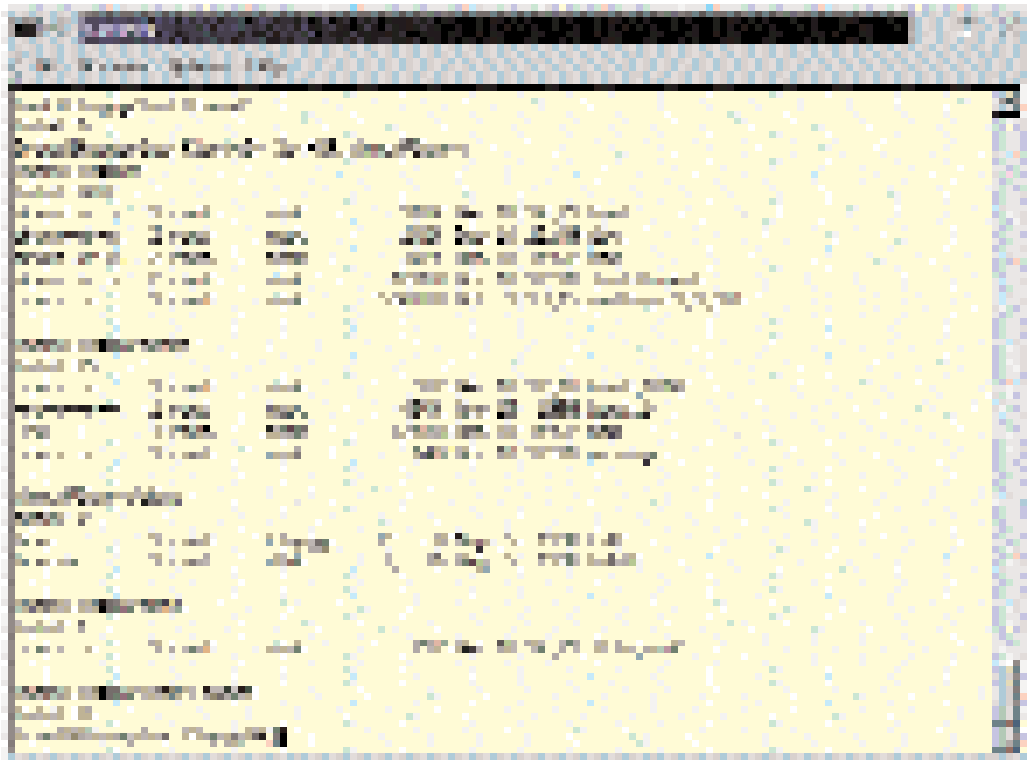

Figure 3 - The contents of a typical boot disk

Creating an Utility disk

The Utility disk is a disk full of extra programs which wouldn't have fitted on the root filesystem, things like such as vi, tar, etc. and maybe programs that reside in */usr/bin and /usr/sbin*. These give you the ability to perform many more activities than would otherwise be the case. (See boxout for example.) Simply follow the steps below, and that´s it!

Insert a blank formatted floppy and type,

```
mkfs -t ext2 /dev/fd0
mount -t ext2 /dev/fd0 /mnt/floppy
cd /mnt/floppy
mkdir bin;mkdir sbin
copy (using cp) the programs you think will b
e useful to these directories.
cd /;umount /mnt/floppy
```

Using the Emergency disk set

On rebooting the machine, follow the steps below,

1. Insert the emergency boot disk and wait for the LILO prompt.
2. At the prompt, you can either boot from the hard disk as normal (if the Linux system isn't broken) or you can type *rescue* to boot from the floppy.
3. After a while a prompt will appear asking you to insert the root filesystem disk. Do so and press enter.
4. Wait for the login prompt and login as root. If you want to use programs off your utility disk, insert it and call:

```
mount -t ext2 /dev/fd0 /usr
```

You can then mount your hard drive filesystems and/or do whatever needs doing.

Figure 4 - The edited config files for the root filesystem

In Conclusion

There are many, many more aspects of the above than can be gone into in a magazine such as this. The essential read is the 'Linux Bootdisk HOWTO', which can usually be found in */usr/doc/HOWTO* or */usr/share/doc/HOWTO* on your system.(Bootdisk-HOWTO.) It contains a large amount of detailed information on this subject and more importantly, what to check if you run into problems.

However, the above should give you a good idea of what's involved and may even help you get a login prompt first time! Good luck. ■

AD
G. Matter