The Loopback Root Filesystem HOWTO

Table of Contents

The Loopback Root Filesystem HOWTO

by Andrew M. Bishop, `amb@gedanken.demon.co.uk`

v1.1, 24 September 1999

This HOWTO explains how to use the Linux loopback device to create a Linux native filesystem format installation that can be run from a DOS partition without re−partitioning. Other uses of this same technique are also discussed.

---

1. Introduction

1.1 Copyright

1.2 Revision History

Version 1.0.0

Initial Version (June 1998)

Version 1.0.1−1.0.3

Slight Modifications, kernel version changes, typos etc. (1998 − July 1999)

Version 1.1

Added Copyright Information and Re−Submitted (September 1999)

2. Principles of Loopback Devices and Ramdisks

First I will describe some of the general principles that are used in the setting up of a loopback filesystem as the root device.

2.1 Loopback Devices

A **loopback device** in Linux is a virtual device that can be used like any other media device.

Examples of normal media devices are hard disk partitions like `/dev/hda1`, `/dev/hda2`, `/dev/sda1`, or entire disks like the floppy disk `/dev/fd0` etc. They are all devices that can be used to hold a files and directory structures. They can be formatted with the filesystem that is required (ext2fs, msdos, ntfs etc.) and then mounted.

The loopback filesystem associates a file on another filesystem as a complete device. This can then be formatted and mounted just like any of the other devices listed above. To do this the device called `/dev/loop0` or `/dev/loop1` etc is associated with the file and then this new virtual device is mounted.

2.2 Ramdisk Devices

In Linux it is also possible to have another type of virtual device mounted as a filesystem, this is the **ramdisk device**.

In this case the device does not refer to any physical hardware, but to a portion of memory that is set aside for the purpose. The memory that is allocated is never swapped out to disk, but remains in the disk cache.

A ramdisk can be created at any time by writing to the ramdisk device `/dev/ram0` or `/dev/ram1` etc. This can then be formatted and mounted in the same way that the loopback device is.

When a ramdisk is used to boot from (as is often done on Linux installation disks or rescue disks) then the disk image (the entire contents of the disk as a single file) can be stored on the boot floppy in a compressed form. This is automatically recognised by the kernel when it boots and is uncompressed into the ramdisk before it is mounted.

2.3 The Initial Ramdisk Device

The **initial ramdisk** device in Linux is another important mechanism that we need to be able to use a loopback device as a the root filesystem.

When the initial ramdisk is used the filesystem image is copied into memory and mounted so that the files on it can be accessed. A program on this ramdisk (called `/linuxrc`) is run and when it is finished a different device is mounted as the root filesystem. The old ramdisk is still present though and is mounted on the directory `/initrd` if present or available through the device `/dev/initrd`.

This is unusual behaviour since the normal boot sequence boots from the designated root partition and keeps on running. With the initial ramdisk option the root partition is allowed to change before the main boot sequence is started.

2.4 The Root Filesystem

The root filesystem is the device that is mounted first so that it appears as the directory called `/` after booting.

There are a number of complications about the root filesystem that are due to the fact that it contains all files. When booting the `rc` scripts are run, these are either the files in `/etc/rc.d` or `/etc/rc?.d` depending on the version of the `/etc/init` program.

When the system has booted it is not possible to unmount the root partition or change it since all programs will be using it to some extent. This is why the initial ramdisk is so useful because it can be used so that the final root partition is not the same as the one that is loaded at boot time.

2.5 The Linux Boot Sequence

To show how the initial ramdisk operates in the boot sequence, the order of events is listed below.

1. The kernel is loaded into memory, this is performed by `LILO` or `LOADLIN`. You can see the `Loading...` message as this happens.
2. The ramdisk image is loaded into memory, again this is performed by `LILO` or `LOADLIN.` You can see the `Loading...` message again as this happens.
3. The kernel is initialised, including parsing the command line options and setting of the ramdisk as the root device.
4. The program `/linuxrc` is run on the initial ramdisk.
5. The root device is changed to that specified in the kernel parameter.
6. The init program `/etc/init` is run which will perform the user configurable boot sequence.

This is just a simplified version of what happens, but is sufficient to explain how the kernel starts up and where the initial ramdisk is used.

---

3. How To Create a Loopback Root Device

Now that the general principles are explained the method of creating the loopback device can be explained.

3.1 Requirements

To create the loopback root device will require a number of things.

- A working Linux system.
- A way to copy large files onto the target DOS partition.

Most important is access to an installed Linux system. This is because the loop device can only be created under Linux. This will mean that it is not possible to bootstrap a working system from nothing. The requirements of the Linux system that you use is that you can compile a kernel on it.

Once the loopback device is created it will be a large file. I have used an 80 MB files, but while this was sufficient for an X terminal it may not be enough if you want to use it for much else. This file must be copied onto the DOS partition, so either a network or a lot of floppy disks must be used.

The software that you will require includes

- `LOADLIN` version 1.6 or above
- A version of `mount` that supports loopback devices
- A version of the kernel that supports the required options.

All of these should be standard for recent Linux installations.

3.2 Creating the Linux Kernel

I created the loopback device using Linux kernel version 2.0.31, other versions should also work, but they must have at least the options listed below.

The kernel options that you will need to enable are the following:

- RAM disk support (`CONFIG_BLK_DEV_RAM`).
- Initial RAM disk (initrd) support (`CONFIG_BLK_DEV_INITRD`).
- Loop device support (`CONFIG_BLK_DEV_LOOP`).
- fat fs support (`CONFIG_FAT_FS`).
- msdos fs support (`CONFIG_MSDOS_FS`).

The first two are for the RAM disk device itself and the initial ram disk device. The next one is the loop back filesystem option. The last two are the msdos filesystem support which is required to mount the DOS partitition.

Compiling a kernel without modules is the easiest option, although if you do want modules then it should be possible although I have not tried it. If modules are used then you should make sure that you have the options above compiled in and not as modules themselves.

Depending on the kernel version that you have you may need to apply a kernel patch. It is a very simple one that allows the loopback device to be used as the root filesystem.

- Kernel versions before 2.0.0; I have no information about these.
- Kernel version 2.0.0 to 2.0.34; you need to apply the kernel patch for 2.0.x kernels as shown below.
- Kernel version 2.0.35 to 2.0.x; no kernel patch is required.
- Kernel version 2.1.x; you need to apply the kernel patch for 2.0.x or 2.2.x kernels as shown below, depending on the exact 2.1.x version.
- Kernel version 2.2.0 to 2.2.10; you need to apply the kernel patch for 2.2.x kernels as shown below.
- Kernel version 2.3.x; you need to apply the kernel patch for 2.2.x kernels as shown below.

For 2.0.x kernels the file `/init/main.c` needs to have a single line added to it as shown by the modified version below. The line that says `"loop", 0x0700` is the one that was added.

```
static void parse_root_dev(char * line)
{
        int base = 0;
        static struct dev_name_struct {
                const char *name;
                const int num;
        } devices[] = {
                { "nfs",     0x00ff },
                { "loop",    0x0700 },
                { "hda",     0x0300 },

...

                { "sonycd",  0x1800 },
                { NULL, 0 }
        };

...

}
```

For 2.2.x kernels the file `/init/main.c` needs to have three lines added to it as shown by the modified version below. The line that says `"loop", 0x0700` and the ones either side of it are the ones that were added.

```
static struct dev_name_struct {
        const char *name;
        const int num;
} root_dev_names[] __initdata = {
#ifdef CONFIG_ROOT_NFS
        { "nfs",     0x00ff },
#endif
#ifdef CONFIG_BLK_DEV_LOOP
        { "loop",    0x0700 },
#endif
#ifdef CONFIG_BLK_DEV_IDE
        { "hda",     0x0300 },

...

        { "ddv", DDV_MAJOR << 8},
#endif
        { NULL, 0 }
};
```

Once the kernel is configured it should be compiled to produce a `zImage` file (`make zImage`). This file will be `arch/i386/boot/zImage` when compiled.

3.3 Creating the Initial Ramdisk Device

The initial ramdisk is most easily created as a loopback device from the start. You will need to do this as root, the commands that you need to execute are listed below, they are assumed to be run from root's home directory (`/root`).

```
mkdir /root/initrd
dd if=/dev/zero of=initrd.img bs=1k count=1024
mke2fs -i 1024 -b 1024 -m 5 -F -v initrd.img
mount initrd.img /root/initrd -t ext2 -o loop
cd initrd
[create the files]
cd ..
umount /root/initrd
gzip -c -9 initrd.img > initrdgz.img
```

There are a number of steps to this, but they can be described as follows.

1. Create a mount point for the initial ramdisk (an empty directory).
2. Create an empty file of the size required. Here I have used 1024kB, you may need less or more depending on the contents, (the size is the last parameter).
3. Make an ext2 filesystem on the empty file.
4. Mount the file onto the mount point, this uses the loopback device.
5. Change to the mounted loopback device.
6. Create the files that are required (see below for details).
7. Move out of the mounted loopback device.
8. Unmount the device.
9. Create a compressed version for use later.

Contents Of The Initial Ramdisk

The files that you will need on the ramdisk are the minimum requirements to be able to execute any commands.

- `/linuxrc` The script that is run to mount the msdos file system (see below).
- `/lib/*` The dynamic linker and the libraries that the programs need.
- `/etc/*` The cache used by the dynamic linker (not strictly needed, but does stop it complaining).
- `/bin/*` A shell interpreter (`ash` because it is smaller than `bash`. The `mount` and `losetup` programs for handling the DOS disk and setting up the loopback devices.
- `/dev/*` The devices that will be used. You need `/dev/zero` for `ld-linux.so`, `/dev/hda*` to mount the msdos disk and `/dev/loop*` for the lopback device.
- `/mnt` An empty directory to mount the msdos disk on.

The initial ramdisk that I used is listed below, the contents come to about 800kB when the overhead of the filesystem are taken into account.

```
total 18
drwxr-xr-x   2 root     root         1024 Jun  2 13:57 bin
drwxr-xr-x   2 root     root         1024 Jun  2 13:47 dev
drwxr-xr-x   2 root     root         1024 May 20 07:43 etc
drwxr-xr-x   2 root     root         1024 May 27 07:57 lib
-rwxr-xr-x   1 root     root          964 Jun  3 08:47 linuxrc
drwxr-xr-x   2 root     root        12288 May 27 08:08 lost+found
drwxr-xr-x   2 root     root         1024 Jun  2 14:16 mnt

./bin:
total 168
-rwxr-xr-x   1 root     root        60880 May 27 07:56 ash
-rwxr-xr-x   1 root     root         5484 May 27 07:56 losetup
-rwsr-xr-x   1 root     root        28216 May 27 07:56 mount
lrwxrwxrwx   1 root     root            3 May 27 08:08 sh -> ash

./dev:
total 0
brw-r--r--   1 root     root       3,   0 May 20 07:43 hda
brw-r--r--   1 root     root       3,   1 May 20 07:43 hda1
brw-r--r--   1 root     root       3,   2 Jun  2 13:46 hda2
brw-r--r--   1 root     root       3,   3 Jun  2 13:46 hda3
brw-r--r--   1 root     root       7,   0 May 20 07:43 loop0
brw-r--r--   1 root     root       7,   1 Jun  2 13:47 loop1
crw-r--r--   1 root     root       1,   3 May 20 07:42 null
crw-r--r--   1 root     root       5,   0 May 20 07:43 tty
crw-r--r--   1 root     root       4,   1 May 20 07:43 tty1
crw-r--r--   1 root     root       1,   5 May 20 07:42 zero

./etc:
total 3
-rw-r--r--   1 root     root         2539 May 20 07:43 ld.so.cache

./lib:
total 649
lrwxrwxrwx   1 root     root           18 May 27 08:08 ld-linux.so.1 -> ld-linux.so.1.7.14
-rwxr-xr-x   1 root     root        21367 May 20 07:44 ld-linux.so.1.7.14
lrwxrwxrwx   1 root     root           14 May 27 08:08 libc.so.5 -> libc.so.5.3.12
-rwxr-xr-x   1 root     root       583795 May 20 07:44 libc.so.5.3.12

./lost+found:
```

```
total 0

./mnt:
total 0
```

The only complex steps about this are the devices in `dev`. Use the `mknod` program to create them, use the existing devices in `/dev` as a template to get the required parameters.

The /linuxrc file

The `/linuxrc` file on the initial ramdisk is required to do all of the preparations so that the loopback device can be used for the root partition when it exits.

The example below tries to mount `/dev/hda1` as an msdos partition and if it succeeds then sets up the files `/linux/linuxdsk.img` as `/dev/loop0` and `/linux/linuxswp.img` as `/dev/loop1`.

```
#!/bin/sh

echo INITRD: Trying to mount /dev/hda1 as msdos

if /bin/mount -n -t msdos /dev/hda1 /mnt; then

   echo INITRD: Mounted OK
   /bin/losetup /dev/loop0 /mnt/linux/linuxdsk.img
   /bin/losetup /dev/loop1 /mnt/linux/linuxswp.img
   exit 0

else

   echo INITRD: Mount failed
   exit 1

fi
```

The first device `/dev/loop0` will become the root device and the second one `/dev/loop1` will become the swap space.

If you want to be able to write to the DOS partition as a non−root user when you have finished then you should use `mount -n -t msdos /dev/hda1 /mnt -o uid=0,gid=0,umask=000,quiet` instead. This will map all accesses to the DOS partition to root and set the permissions appropriately.

3.4 Creating The Root Device

The root device that you will be using is the file `linuxdsk.img`. You will need to create this in the same way that the initial ramdisk was created, but bigger. You can install any Linux installation that you like onto this disk.

The easiest way might be to copy an existing Linux installation into it. An alternative is to install a new Linux installation onto it.

Assuming that you have done this, there are some minor changes that you must make.

The `/etc/fstab` file must reference the root partition and the swap using the two loopback devices that are setup on the initial ramdisk.

```
/dev/loop0     /      ext2   defaults 1 1
/dev/loop1     swap   swap   defaults 1 1
```

This will ensure that when the real root device is used the kernel will not be confused about where the root device is. It will also allow the swap space to be added in the same way a swap partition is normally used. You should remove any other reference to a root disk device or swap partition.

If you want to be able to read the DOS partition after Linux has started then you will need to make a number of extra small changes.

Create a directory called `/initrd`, this is where the initial ramdisk will be mounted once the loopback root filesystem is mounted.

Create a symbolic link called `/DOS` that points to `/initrd/mnt` where the real DOS parition will be mounted.

Add a line into the rc file that mounts the disks. This should run the command `mount -f -t msdos /dev/hda1 /initrd/mnt`, this will create a 'fake' mount of the DOS partition so that all programs (like `df`) will know that the DOS partition is mounted and where to find it. If you used different options in the `/linuxrc` file that obviously you should use them here also.

There is no need to have a Linux kernel on this root device since that is already loaded earlier. If you are using modules however then you should include them on this device as normal.

3.5 Creating the Swap Device

The root device that you will be using is the file `linuxswap.img`. The swap device is very simple to create. Create an empty file as was done for the initial ramdisk and then run `mkswap linuxswap.img` to intialise it.

The size of the swap space will depend on what you plan to do with the installed system, but I would recommend between 8MB and the amount of RAM that you have.

3.6 Creating the MSDOS Directory

The files that are going to be used need to be moved onto the DOS partition.

The files that are required in the DOS directory called `C:\LINUX` are the following:

- `LINUXDSK.IMG` The disk image that will become the root device.
- `LINUXSWP.IMG` The swap space.

3.7 Creating the Boot Floppy

The boot floppy that is used is just a normal DOS format bootable floppy.

This is created using `format a: /s` from DOS.

Onto this disk you will need to create an `AUTOEXEC.BAT` file (as below) and copy the kernel, compressed initial ramdisk and `LOADLIN` executable.

- `AUTOEXEC.BAT` The DOS automatically executed batch file.
- `LOADLIN.EXE` The `LOADLIN` program executable.
- `ZIMAGE` The Linux kernel.
- `INITRDGZ.IMG` The compressed initial ramdisk image.

The `AUTOEXEC.BAT` file should contain just one line as below.

```
\loadlin \zImage initrd=\initrdgz.img root=/dev/loop0 ro
```

This specifies the kernel image to use, the initial ramdisk image, the root device after the initial ramdisk has finished and that the root partition is to be mounted read–only.

4. Booting the System

To boot from this new root device all that is required is that the floppy disk prepared as described above is inserted for the PC to boot from.

You will see the following sequence of events.

1. DOS boots
2. AUTOEXEC.BAT starts
3. LOADLIN is run
4. The Linux kernel is copied into memory
5. The initial ramdisk is copied into memory
6. The Linux kernel is started running
7. The `/linuxrc` file on the initial ramdisk is run
8. The DOS partition is mounted and the root and swap devices set up
9. The boot sequence continues from the loopback device

When this is complete you can remove the boot floppy and use the Linux system.

4.1 Possible Problems With Solutions

There are a number of stages where this process could fail, I will try to explain what they are and what to check.

DOS booting is easy to recognise by the message that it prints `MS−DOS Starting ...` on the screen. If this is not seen then the floopy disk is either not–bootable or the PC is not bootable from the floppy disk drive.

When the `AUTOEXEC.BAT` file is run the commands in it should be echoed to the screen by default. In this case there is just the single line in the file that starts `LOADLIN`.

When LOADLIN executes it will do two very visible things, firstly it will load the kernel into memory, secondly it will copy the ramdisk into memory. Both of these are indicated by a `Loading...` message.

The kernel starts by uncompressing itself, this can give **crc** errors if the kernel image is corrupted. Then it will start running the initialisation sequence which is very verbose with diagnostic messages. Loading of the initial ramdisk device is also visible during this phase.

When the `/linuxrc` file is run there is no diagnostic messages, but you can add these yourself as an aid to debugging. If this stage fails to set up the loopback device as the root device then you may see a message that there is no root device and the kernel aborts.

The normal boot sequence of the new root device will now continue and this is quite verbose. There may be problems about the root device being mounted read−write, but the LOADLIN command line option '`ro`' should stop that. Other problems that can occur are that the boot sequence is confused about where the root device is, this is probably due to a problem with `/etc/fstab`.

When the boot sequence has completed, the remaining problem is that programs are confused about whether the DOS partition is mounted or not. This is why it is a good idea to use the fake mount command described earlier. This makes life a lot easier if you want to access the files on the DOS device.

4.2 Reference Documents

The documents that I used to create my first loopback root filesystem were:

- The Linux kernel source, in particular `init/main.c`
- The Linux kernel documentation, in particular `Documentation/initrd.txt` and `Documentation/ramdisk.txt`.
- The LILO documentation.
- The LOADLIN documentation

---

5. Other Loopback Root Device Possibilities

Once the principle of booting a filesystem in a file on a DOS partition has been established there are many other things that you can now do.

5.1 DOS Hard−disk Only Installation

If it is possible to boot Linux from a file on a DOS harddisk by using a boot floppy then it is obviously also possible to do it using the harddisk itself.

A configuration boot menu can be used to give the option of running LOADLIN from within the AUTOEXEC.BAT. This will give a much faster boot sequence, but is otherwise identical.

5.2 LILO Booted Installation

Using LOADLIN is only one option for booting a Linux kernel. There is also LILO that does much the same but without needing DOS.

In this case the DOS format floppy disk can be replaced by an ext2fs format one. Otherwise the details are very similar, with the kernel and the initial ramdisk being files on that disk.

The reason that I chose the `LOADLIN` method is that the arguments that need to be given to `LILO` are slightly more complex. Also it is more obvious to a casual observer what the floppy disk is since it can be read under DOS.

5.3 VFAT / NTFS Installation

I have tried the NTFS method, and have had no problems with it. The NTFS filesystem driver is not a standard kernel option in version 2.0.x, but is available as a patch from http://www.informatik.hu−berlin.de/~loewis/ntfs/. In version 2.2.x the NTFS driver is included as standard in the kernel.

The only changes for the VFAT or NTFS options are in the initial ramdisk, the file `/linuxrc` needs to mount a file system of type vfat or ntfs rather that msdos.

I know of no reason why this should not also work on a VFAT partition.

5.4 Installing Linux without Re−partitioning

The process of installing Linux on a PC from a standard distribution requires booting from a floppy disk and re−partitioning the disk. This stage could instead be accomplished by a boot floppy that creates an empty loopback device and swap file. This would allow the installation to proceed as normal, but it would install into the loopback device rather than a partition.

This could be used as an alternative to a `UMSDOS` installation, it would be more efficient in disk usage since the minimum allocation unit in the ext2 filesystem is 1kB instead of up to 32kB on a DOS partition. It can also be used on VFAT and NTFS formatted disks which are otherwise a problem.

5.5 Booting From a Non−bootable device

This method can also be used to boot a Linux system from a device that is not normally bootable.

- CD−Rom
- Zip Disks
- Parallel port disk drives

Obviously there are many other devices that could be used, NFS root filesystems are already included in the kernel as an option, but the method described here might also be used instead.