Configuring the Linux kernel with initrd support

Andreas Fester

March 3, 2004

Abstract

This article describes the concept of loading linux with an initial ramdisk. Using this approach, the static part of the kernel can be as small as possible, because almost all drivers (even drivers which are needed to access the hard drive) can be loaded as modules.

1 Introduction

This is version 0.1 of this document.

This article applies to kernel 2.6.x and the Debian Linux distribution. With other kernel versions or other distributions some of the steps might differ. Also, the article is not complete at all; it is just a very first approach to describe the initrd mechanism!

2 Step-by-Step tutorial

2.1 Step 1: Compiling the kernel

The first step is to compile the kernel with the proper options set. Since most kernel components can be loaded as modules when using initrd, most configuration parameters should be set to "module".

Additionally, make sure that at least the following configuration parameters are enabled:

- Block Devices / Ram Disk Support (*BLK_DEV_RAM*) must be compiled into the kernel (not as module)
- Block Devices / Ram Disk Support / Initial RAM disk (initrd) support (*BLK_DEV_INITRD*) must be enabled
- Files Systems / Miscellaneous filesystems / Compressed ROM File System Support (*CRAMFS*) must be compiled into the kernel (not as module)

2.2 Step 2: Creating the ramdisk image

When using an initial ramdisk, the boot loader loads the kernel and the ramdisk image into RAM and mounts the ramdisk image read-write as "/". Therefore, the next step is to create this ramdisk image. The command **mkinitrd** can be used for this. Basically, the image must contain any modules which need to be loaded in order to mount the root filesystem. These are usually the low level hardware driver (e.g. ide) and the driver for the file system (e.g. ext2). To use **mkinitrd**, simply call it with the **-o** option which names the output file. Example:

```
% cd /tmp
% mkinitrd -o initrd.img
% ls -la initrd.img
-rw-r--r--    1 root     root      1720320 2004-03-20 15:31 initrd.img
% file initrd.img
initrd.img: Linux Compressed ROM File System data, little endian size 1720320
version #2 sorted_dirs CRC 0xf37d202e, edition 0, 888 blocks, 102 files
```

You can now mount the image (provided that you have loopback device support in your kernel) and list its contents:

```
% mount -o loop initrd.img /mnt/tmp
% cd /mnt/tmp
% ls -la
total 9
drwxr-xr-x    1 root     root     124 1970-01-01 01:00 bin
drwxr-xr-x    1 root     root     104 1970-01-01 01:00 dev
drwxr-xr-x    1 root     root       0 1970-01-01 01:00 dev2
drwxr-xr-x    1 root     root       0 1970-01-01 01:00 devfs
drwxr-xr-x    1 root     root      28 1970-01-01 01:00 etc
drwxr-xr-x    1 root     root      64 1970-01-01 01:00 lib
-rwxr-xr-x    1 root     root     292 1970-01-01 01:00 linuxrc
-rw-r--r--    1 root     root      55 1970-01-01 01:00 linuxrc.conf
-rw-r--r--    1 root     root      53 1970-01-01 01:00 loadmodules
drwxr-xr-x    1 root     root       0 1970-01-01 01:00 mnt
drwxr-xr-x    1 root     root       0 1970-01-01 01:00 proc
drwxr-xr-x    1 root     root      60 1970-01-01 01:00 sbin
-rw-r--r--    1 root     root       0 1970-01-01 01:00 script
drwxr-xr-x    1 root     root       0 1970-01-01 01:00 scripts
drwxr-xr-x    1 root     root       0 1970-01-01 01:00 sys
drwxr-xr-x    1 root     root       0 1970-01-01 01:00 tmp
drwxr-xr-x    1 root     root      16 1970-01-01 01:00 usr
drwxr-xr-x    1 root     root       0 1970-01-01 01:00 var
```

As you can see, the image contains a complete root filesystem with all necessary directories. Most of them are empty (`var`, `tmp`, `sys`, `scripts`, `proc`, `mnt`, `devfs`, `dev2`). The others contain the minimal set of programs necessary to mount the root filesystem and continue the boot process. For example, the bin directory contains the mount command which is dynamically linked against the C library and the dynamic loader library. Both of them are of course also available in the ramdisk, because they are needed to execute the mount command. The lib/modules - directory contains the kernel modules necessary to mount the root image.

How **mkinitrd** creates the ramdisk image can be configured through some configuration files which are stored in `/etc/mkinitrd`. The file `mkinitrd.conf` contains the main configuration options, while the file `modules` contains a list of modules to load and their parameters. For now, you should start with the default settings which should create a fairly usable image.

2.3 Step 3: Installing the kernel

Provided that you already have a running linux system, you should leave it so. That is, install the new initrd based kernel as an optional kernel which can be choosen with your boot loader. Otherwise, if anything fails when booting the new kernel, you might have big trouble. Once the initrd based kernel boots, it can be configured to be the primary (or one and only) kernel to boot.

Copy the kernel image to the appropriate place, e.g. **cp arch/i386/boot/bzImage /boot/vmlinuz-initrd**. Then, create a new entry in your boot loader. For example, for Lilo, I added

```
image  = /boot/vmlinuz-initrd
label  = InitRd
root   = /dev/hda14
initrd = /boot/initrd.img
```

If you are usually using a special *vga=* setting, you should remove it, otherwise if some modules are missing you could end up with a black screen and can not see what is happening.

This means of course that **mkinitrd** has been used before to create the ramdisk image in this directory:

```
% cd /boot
% mkinitrd -o initrd.img
```

Now run lilo to add the new kernel. Keep fingers crossed and reboot :-)

3 Fine tuning

Fine tuning among others means to make the image as small as possible by removing unnecessary components.

Links to useful resources

URLs

[1] Andries Brouwer, *util-linux: Miscellaneous utilities for Linux* <ftp://ftp.win.tue.nl/pub/linux-local/utils

[2] Werner Almesberger, *Booting Linux: The History and the Futurex* <ftp://icaftp.epfl.ch/pub/people/almesber/booting/bootinglinux-current.ps.gz> *(currently offline)* .

[3] , *The file initrd.txt in the Documentation directory of the linux kernel tree provides a more detailed description of the boot process.*.

[4] , *newlib package (experimental), with initrd example* <ftp://icaftp.epfl.ch/pub/people/almesber/misc/new *(currently offline)* .