Linux Ramdisk mini-HOWTO

by Van Emery

Introduction

What is a RAM disk? A RAM disk is a portion of RAM which is being used as if it were a disk drive. RAM disks have fixed sizes, and act like regular disk partitions. Access time is much faster for a RAM disk than for a real, physical disk. However, any data stored on a RAM disk is lost when the system is shut down or powered off. RAM disks can be a great place to store temporary data.

The Linux kernel version 2.4 has built-in support for ramdisks. Ramdisks are useful for a number of things, including:

- Working with the unencrypted data from encrypted documents
- Serving certain types of web content
- Mounting Loopback file systems (such as run-from-floppy/CD distributions)

Why did I write this document? Because I needed to setup a 16 MB ramdisk for viewing and creating encrypted documents. I did not want the unencrypted documents to be written to any physical media on my workstation. I also found it amazing that I could easily create a "virtual disk" in RAM that is larger than my first hard drive, a 20 MB Winchester disk. At the time, that disk was so large that I never even considered filling it up, and I never did!

This document should take you step-by-step through the process of creating and using RAM disks.

Assumptions/Setup

I was using Red Hat 9 for this test, but it should work with other GNU/Linux distributions running 2.4.x kernels. I am also assuming that the distribution you are using already has ramdisk support compiled into the kernel. My test machine was a Pentium 4 and had 256 MB of RAM. The exact version of the kernel that I used was: 2.4.20-20.9

Step 1: Take a look at what has already been created by your system

Red Hat creates 16 ramdisks by default, although they are not "active" or using any RAM. It lists devices ram0 - ram 19, but only ram0 - ram15 are usable by default. To check these block devices out, use the following command:

```
[root]# ls -l /dev/ram*
lrwxrwxrwx    1 root     root            4 Jun 12 00:31 /dev/ram -> ram1
brw-rw----    1 root     disk       1,   0 Jan 30  2003 /dev/ram0
brw-rw----    1 root     disk       1,   1 Jan 30  2003 /dev/ram1
brw-rw----    1 root     disk       1,  10 Jan 30  2003 /dev/ram10
brw-rw----    1 root     disk       1,  11 Jan 30  2003 /dev/ram11
brw-rw----    1 root     disk       1,  12 Jan 30  2003 /dev/ram12
brw-rw----    1 root     disk       1,  13 Jan 30  2003 /dev/ram13
brw-rw----    1 root     disk       1,  14 Jan 30  2003 /dev/ram14
brw-rw----    1 root     disk       1,  15 Jan 30  2003 /dev/ram15
brw-rw----    1 root     disk       1,  16 Jan 30  2003 /dev/ram16
brw-rw----    1 root     disk       1,  17 Jan 30  2003 /dev/ram17
brw-rw----    1 root     disk       1,  18 Jan 30  2003 /dev/ram18
brw-rw----    1 root     disk       1,  19 Jan 30  2003 /dev/ram19
brw-rw----    1 root     disk       1,   2 Jan 30  2003 /dev/ram2
brw-rw----    1 root     disk       1,   3 Jan 30  2003 /dev/ram3
brw-rw----    1 root     disk       1,   4 Jan 30  2003 /dev/ram4
brw-rw----    1 root     disk       1,   5 Jan 30  2003 /dev/ram5
brw-rw----    1 root     disk       1,   6 Jan 30  2003 /dev/ram6
brw-rw----    1 root     disk       1,   7 Jan 30  2003 /dev/ram7
brw-rw----    1 root     disk       1,   8 Jan 30  2003 /dev/ram8
brw-rw----    1 root     disk       1,   9 Jan 30  2003 /dev/ram9
lrwxrwxrwx    1 root     root            4 Jun 12 00:31 /dev/ramdisk -> ram0
```

Now, grep through dmesg output to find out what size the ramdisks are:

```
[root]# dmesg | grep RAMDISK
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
RAMDISK: Compressed image found at block 0
```

As you can see, the default ramdisk size is 4 MB. I want a 16 MB ramdisk, so the next step will be to configure Linux to use a larger ramdisk size during boot.

Step 2: Increase ramdisk size

Ramdisk size is controlled by a command-line option that is passed to the kernel during boot. Since GRUB is the default bootloader for Red Hat 9, I will modify `/etc/grub.conf` with the new kernel option. The kernel option for ramdisk size is: `ramdisk_size=xxxxx`, where `xxxxx` is the size expressed in 1024-byte blocks. Here is what I will add to `/etc/grub.conf` to configure 16 MB ramdisks:

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/hda5
#          initrd /initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Linux (2.4.20-20.9)
        root (hd0,0)
        kernel /vmlinuz-2.4.20-20.9 ro root=LABEL=/ hdc=ide-scsi ramdisk_size=16000
        initrd /initrd-2.4.20-20.9.img
```

Once you save the file, you will need to reboot your system. After the reboot, a look at the

dmesg output should confirm the change has taken effect:

```
[root]# dmesg | grep RAMDISK
RAMDISK driver initialized: 16 RAM disks of 16000K size 1024 blocksize
RAMDISK: Compressed image found at block 0
```

Step 3: Format the ramdisk

There is no need to format the ramdisk as a journaling file system, so we will simply use the ubiquitous ext2 file system. I only want to use one ramdisk, so I will only format `/dev/ram0`:

```
[root]# mke2fs -m 0 /dev/ram0
mke2fs 1.32 (09-Nov-2002)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
4000 inodes, 16000 blocks
0 blocks (0.00%) reserved for the super user
First data block=1
2 block groups
8192 blocks per group, 8192 fragments per group
2000 inodes per group
Superblock backups stored on blocks:
        8193

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 22 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

The `-m 0` option keeps mke2fs from reserving any space on the file system for the root user, which is the default behavior. I want all of the ramdisk space available to a regular user for working with encrypted files.

Step 4: Create a mount point and mount the ramdisk

Now that you have formatted the ramdisk, you must create a mount point for it. Then you can mount your ramdisk and use it. We will use the directory `/mnt/rd` for this operation.

```
[root]# mkdir /mnt/rd
[root]# mount /dev/ram0 /mnt/rd
```

Now verify the new ramdisk mount:

```
[root]# mount | grep ram0
/dev/ram0 on /mnt/rd type ext2 (rw)
[root]# df -h | grep ram0
/dev/ram0              16M   13K   16M   1% /mnt/rd
```

You can even take a detailed look at the new ramdisk with the tune2fs command:

```
[root]# tune2fs -l /dev/ram0
tune2fs 1.32 (09-Nov-2002)
Filesystem volume name:   none
Last mounted on:          not available
Filesystem UUID:          fbb80e9a-8e7c-4bd4-b3d9-37c29813a5f5
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
```

```
Filesystem features:      filetype sparse_super
Default mount options:    (none)
Filesystem state:         not clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              4000
Block count:              16000
Reserved block count:     0
Free blocks:              15478
Free inodes:              3989
First block:              1
Block size:               1024
Fragment size:            1024
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         2000
Inode blocks per group:   250
Filesystem created:       Mon Dec  8 14:33:57 2003
Last mount time:          Mon Dec  8 14:35:39 2003
Last write time:          Mon Dec  8 14:35:39 2003
Mount count:              1
Maximum mount count:      22
Last checked:             Mon Dec  8 14:33:57 2003
Check interval:           15552000 (6 months)
Next check after:         Sat Jun  5 14:33:57 2004
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
```

In my case, I need the user "van" to be able to read and write to the ramdisk, so I must change the ownership and permissions of the `/mnt/rd` directory:

```
[root]# chown van:root /mnt/rd
[root]# chmod 0770 /mnt/rd
[root]# ls -ald /mnt/rd
drwxrwx---    2 van     root         4096 Dec  8 11:09 /mnt/rd
```

The ownership and permissions on the ramdisk filesystem/directory should be tailored to your particular needs.

Step 5: Use the ramdisk

Now that it has been created, you can copy, move, delete, edit, and list files on the ramdisk exactly as if they were on a physical disk partiton. This is a great place to view decrypted GPG or OpenSSL files, as well as a good place to create files that will be encrypted. After your host is powered down, all traces of files created on the ramdisk are gone.

To unmount the ramdisk, simply enter the following:

```
[root]# umount -v /mnt/rd
/dev/ram0 umounted
```

Note: If you remount the ramdisk, your data will still be there. Once memory has been allocated to the ramdisk, it is flagged so that the kernel will not try to reuse the memory later. Therefore, you cannot "reclaim" the RAM after you are done with using the ramdisk. For this reason, you will want to be careful not to allocate more memory to the ramdisk than is absolutely necessary. In my case, I am allocating < 10% of the physical RAM. You will have to tailor the ramdisk size to your needs. Of course, you can always free up the space with a reboot!

Automating Ramdisk Creation

If you need to create and mount a ramdisk every time your system boots, you can automate the process by adding some commands to your `/etc/rc.local` init script. Here are the lines that I added:

```
# Formats, mounts, and sets permissions on my 16MB ramdisk
/sbin/mke2fs -q -m 0 /dev/ram0
/bin/mount /dev/ram0 /mnt/rd
/bin/chown van:root /mnt/rd
/bin/chmod 0750 /mnt/rd
```

Conclusion

You have now seen how to setup and use a ramdisk on your GNU/Linux system. Hopefully, you will find this information to be interesting and useful!

Resources

- /usr/src/linux-2.4/Documentation/ramdisk.txt
- /usr/src/linux-2.4/drivers/block/rd.c
- `man mke2fs`
- Ramdisk article by Mark Nielsen for Red Hat 6.0

Back to Linux Gouge...