Using the RAM disk block device with Linux

Contents:

1) Overview

The RAM disk driver is a way to use main system memory as a block device. It is required for initrd, an initial filesystem used if you need to load modules in order to access the root filesystem (see Documentation/initrd.txt). It can also be used for a temporary filesystem for crypto work, since the contents are erased on reboot.

The RAM disk dynamically grows as more space is required. It does this by using RAM from the buffer cache. The driver marks the buffers it is using as dirty so that the VM subsystem does not try to reclaim them later.

The RAM disk supports up to 16 RAM disks by default, and can be reconfigured to support an unlimited number of RAM disks (at your own risk). Just change the configuration symbol BLK_DEV_RAM_COUNT in the Block drivers config menu and (re)build the kernel.

To use RAM disk support with your system, run './MAKEDEV ram' from the /dev directory. RAM disks are all major number 1, and start with minor number 0 for /dev/ram0, etc. If used, modern kernels use /dev/ram0 for an initrd.

The new RAM disk also has the ability to load compressed RAM disk images, allowing one to squeeze more programs onto an average installation or rescue floppy disk.

2) Kernel Command Line Parameters

ramdisk_size=N

This parameter tells the RAM disk driver to set up RAM disks of N k size. The default is 4096 (4 MB) (8192 (8 MB) on S390).

ramdisk_blocksize=N

This parameter tells the RAM disk driver how many bytes to use per block. The default is 1024 (BLOCK_SIZE).

3) Using "rdev -r"

The usage of the word (two bytes) that "rdev -r" sets in the kernel image is as follows. The low 11 bits (0 -> 10) specify an offset (in 1 k blocks) of up to 2 MB (2^11) of where to find the RAM disk (this used to be the size). Bit 14 indicates that a RAM disk is to be loaded, and bit 15 indicates whether a prompt/wait sequence is to be given before trying to read the RAM disk. Since the RAM disk dynamically grows as data is being written into it, a size field is not required. Bits 11 to 13 are not currently used and may as well be zero. These numbers are no magical secrets, as seen below:

```
./arch/i386/kernel/setup.c:#define RAMDISK_IMAGE_START_MASK     0x07FF
./arch/i386/kernel/setup.c:#define RAMDISK_PROMPT_FLAG          0x8000
./arch/i386/kernel/setup.c:#define RAMDISK_LOAD_FLAG            0x4000
```

Consider a typical two floppy disk setup, where you will have the kernel on disk one, and have already put a RAM disk image onto disk #2.

Hence you want to set bits 0 to 13 as 0, meaning that your RAM disk starts at an offset of 0 kB from the beginning of the floppy.
The command line equivalent is: "ramdisk_start=0"

You want bit 14 as one, indicating that a RAM disk is to be loaded.
The command line equivalent is: "load_ramdisk=1"

You want bit 15 as one, indicating that you want a prompt/keypress sequence so that you have a chance to switch floppy disks.
The command line equivalent is: "prompt_ramdisk=1"

Putting that together gives 2^15 + 2^14 + 0 = 49152 for an rdev word.
So to create disk one of the set, you would do:

```
        /usr/src/linux# cat arch/i386/boot/zImage > /dev/fd0
        /usr/src/linux# rdev /dev/fd0 /dev/fd0
        /usr/src/linux# rdev -r /dev/fd0 49152
```

If you make a boot disk that has LILO, then for the above, you would use:
```
        append = "ramdisk_start=0 load_ramdisk=1 prompt_ramdisk=1"
```
Since the default start = 0 and the default prompt = 1, you could use:
```
        append = "load_ramdisk=1"
```

4) An Example of Creating a Compressed RAM Disk

To create a RAM disk image, you will need a spare block device to construct it on. This can be the RAM disk device itself, or an unused disk partition (such as an unmounted swap partition). For this example, we will use the RAM disk device, "/dev/ram0".

Note: This technique should not be done on a machine with less than 8 MB of RAM. If using a spare disk partition instead of /dev/ram0, then this restriction does not apply.

a) Decide on the RAM disk size that you want. Say 2 MB for this example.
   Create it by writing to the RAM disk device. (This step is not currently
   required, but may be in the future.) It is wise to zero out the

```
   area (esp. for disks) so that maximal compression is achieved for
   the unused blocks of the image that you are about to create.

        dd if=/dev/zero of=/dev/ram0 bs=1k count=2048

b) Make a filesystem on it. Say ext2fs for this example.

        mke2fs -vm0 /dev/ram0 2048

c) Mount it, copy the files you want to it (eg: /etc/* /dev/* ...)
   and unmount it again.

d) Compress the contents of the RAM disk. The level of compression
   will be approximately 50% of the space used by the files. Unused
   space on the RAM disk will compress to almost nothing.

        dd if=/dev/ram0 bs=1k count=2048 | gzip -v9 > /tmp/ram_image.gz

e) Put the kernel onto the floppy

        dd if=zImage of=/dev/fd0 bs=1k

f) Put the RAM disk image onto the floppy, after the kernel. Use an offset
   that is slightly larger than the kernel, so that you can put another
   (possibly larger) kernel onto the same floppy later without overlapping
   the RAM disk image. An offset of 400 kB for kernels about 350 kB in
   size would be reasonable. Make sure offset+size of ram_image.gz is
   not larger than the total space on your floppy (usually 1440 kB).

        dd if=/tmp/ram_image.gz of=/dev/fd0 bs=1k seek=400

g) Use "rdev" to set the boot device, RAM disk offset, prompt flag, etc.
   For prompt_ramdisk=1, load_ramdisk=1, ramdisk_start=400, one would
   have 2^15 + 2^14 + 400 = 49552.

        rdev /dev/fd0 /dev/fd0
        rdev -r /dev/fd0 49552

That is it. You now have your boot/root compressed RAM disk floppy. Some
users may wish to combine steps (d) and (f) by using a pipe.

--------------------------------------------------------------------------
                                                Paul Gortmaker 12/95

Changelog:
----------

10-22-04 :      Updated to reflect changes in command line options, remove
                obsolete references, general cleanup.
                James Nelson (james4765@gmail.com)


12-95 :         Original Document
```