Red Hat Enterprise Linux Release 9.2 Manual Pages on 'initrd.4' command

$ man initrd.4

INITRD(4)　　　　　Linux Programmer's Manual　　　　　INITRD(4)

NAME

   initrd - boot loader initialized RAM disk

CONFIGURATION

   /dev/initrd is a read-only block device assigned major number 1 and mi?

   nor number 250. Typically /dev/initrd is owned by root:disk with mode

   0400 (read access by root only). If the Linux system does not have

   /dev/initrd already created, it can be created with the following com?

   mands:

      mknod -m 400 /dev/initrd b 1 250

      chown root:disk /dev/initrd

   Also, support for both "RAM disk" and "Initial RAM disk" (e.g., CON?

   FIG_BLK_DEV_RAM=y and CONFIG_BLK_DEV_INITRD=y) must be compiled di?

   rectly into the Linux kernel to use /dev/initrd. When using /dev/ini?

   trd, the RAM disk driver cannot be loaded as a module.

DESCRIPTION

   The special file /dev/initrd is a read-only block device. This device

   is a RAM disk that is initialized (e.g., loaded) by the boot loader be?

   fore the kernel is started. The kernel then can use /dev/initrd's con?

   tents for a two-phase system boot-up.

   In the first boot-up phase, the kernel starts up and mounts an initial

   root filesystem from the contents of /dev/initrd (e.g., RAM disk ini?

   tialized by the boot loader). In the second phase, additional drivers

or other modules are loaded from the initial root device's contents.
After loading the additional modules, a new root filesystem (i.e., the
normal root filesystem) is mounted from a different device.

Boot-up operation

When booting up with initrd, the system boots as follows:

1. The boot loader loads the kernel program and /dev/initrd's contents
   into memory.

2. On kernel startup, the kernel uncompresses and copies the contents
   of the device /dev/initrd onto device /dev/ram0 and then frees the
   memory used by /dev/initrd.

3. The kernel then read-write mounts the device /dev/ram0 as the ini‐
   tial root filesystem.

4. If the indicated normal root filesystem is also the initial root
   filesystem (e.g., /dev/ram0) then the kernel skips to the last step
   for the usual boot sequence.

5. If the executable file /linuxrc is present in the initial root
   filesystem, /linuxrc is executed with UID 0. (The file /linuxrc
   must have executable permission. The file /linuxrc can be any valid
   executable, including a shell script.)

6. If /linuxrc is not executed or when /linuxrc terminates, the normal
   root filesystem is mounted. (If /linuxrc exits with any filesystems
   mounted on the initial root filesystem, then the behavior of the
   kernel is UNSPECIFIED. See the NOTES section for the current kernel
   behavior.)

7. If the normal root filesystem has a directory /initrd, the device
   /dev/ram0 is moved from / to /initrd. Otherwise, if the directory
   /initrd does not exist, the device /dev/ram0 is unmounted. (When
   moved from / to /initrd, /dev/ram0 is not unmounted and therefore
   processes can remain running from /dev/ram0. If directory /initrd
   does not exist on the normal root filesystem and any processes re‐
   main running from /dev/ram0 when /linuxrc exits, the behavior of the
   kernel is UNSPECIFIED. See the NOTES section for the current kernel
   behavior.)

8. The usual boot sequence (e.g., invocation of /sbin/init) is per?

   formed on the normal root filesystem.

Options

   The following boot loader options, when used with initrd, affect the

   kernel's boot-up operation:

   initrd=filename

      Specifies the file to load as the contents of /dev/initrd. For

      LOADLIN this is a command-line option. For LILO you have to use

      this command in the LILO configuration file /etc/lilo.config.

      The filename specified with this option will typically be a

      gzipped filesystem image.

   noinitrd

      This boot option disables the two-phase boot-up operation. The

      kernel performs the usual boot sequence as if /dev/initrd was

      not initialized. With this option, any contents of /dev/initrd

      loaded into memory by the boot loader contents are preserved.

      This option permits the contents of /dev/initrd to be any data

      and need not be limited to a filesystem image. However, device

      /dev/initrd is read-only and can be read only one time after

      system startup.

   root=device-name

      Specifies the device to be used as the normal root filesystem.

      For LOADLIN this is a command-line option. For LILO this is a

      boot time option or can be used as an option line in the LILO

      configuration file /etc/lilo.config. The device specified by

      this option must be a mountable device having a suitable root

      filesystem.

Changing the normal root filesystem

   By default, the kernel's settings (e.g., set in the kernel file with

   rdev(8) or compiled into the kernel file), or the boot loader option

   setting is used for the normal root filesystems. For an NFS-mounted

   normal root filesystem, one has to use the nfs_root_name and

   nfs_root_addrs boot options to give the NFS settings. For more infor?

mation on NFS-mounted root see the kernel documentation file Documenta?
tion/filesystems/nfs/nfsroot.txt (or Documentation/filesystems/nfs?
root.txt before Linux 2.6.33). For more information on setting the
root filesystem see also the LILO and LOADLIN documentation.

It is also possible for the /linuxrc executable to change the normal
root device. For /linuxrc to change the normal root device, /proc must
be mounted. After mounting /proc, /linuxrc changes the normal root de?
vice by writing into the proc files /proc/sys/kernel/real-root-dev,
/proc/sys/kernel/nfs-root-name, and /proc/sys/kernel/nfs-root-addrs.
For a physical root device, the root device is changed by having /lin?
uxrc write the new root filesystem device number into /proc/sys/ker?
nel/real-root-dev. For an NFS root filesystem, the root device is
changed by having /linuxrc write the NFS setting into files
/proc/sys/kernel/nfs-root-name and /proc/sys/kernel/nfs-root-addrs and
then writing 0xff (e.g., the pseudo-NFS-device number) into file
/proc/sys/kernel/real-root-dev. For example, the following shell com?
mand line would change the normal root device to /dev/hdb1:

    echo 0x365 >/proc/sys/kernel/real-root-dev

For an NFS example, the following shell command lines would change the
normal root device to the NFS directory /var/nfsroot on a local net?
worked NFS server with IP number 193.8.232.7 for a system with IP num?
ber 193.8.232.2 and named "idefix":

    echo /var/nfsroot >/proc/sys/kernel/nfs-root-name

    echo 193.8.232.2:193.8.232.7::255.255.255.0:idefix \

       >/proc/sys/kernel/nfs-root-addrs

    echo 255 >/proc/sys/kernel/real-root-dev

Note: The use of /proc/sys/kernel/real-root-dev to change the root
filesystem is obsolete. See the Linux kernel source file Documenta?
tion/admin-guide/initrd.rst (or Documentation/initrd.txt before Linux
4.10) as well as pivot_root(2) and pivot_root(8) for information on the
modern method of changing the root filesystem.

Usage

The main motivation for implementing initrd was to allow for modular

kernel configuration at system installation.

A possible system installation scenario is as follows:

1. The loader program boots from floppy or other media with a minimal kernel (e.g., support for /dev/ram, /dev/initrd, and the ext2 filesystem) and loads /dev/initrd with a gzipped version of the ini? tial filesystem.

2. The executable /linuxrc determines what is needed to (1) mount the normal root filesystem (i.e., device type, device drivers, filesys? tem) and (2) the distribution media (e.g., CD-ROM, network, tape, ...). This can be done by asking the user, by auto-probing, or by using a hybrid approach.

3. The executable /linuxrc loads the necessary modules from the initial root filesystem.

4. The executable /linuxrc creates and populates the root filesystem. (At this stage the normal root filesystem does not have to be a com? pleted system yet.)

5. The executable /linuxrc sets /proc/sys/kernel/real-root-dev, un? mounts /proc, the normal root filesystem and any other filesystems it has mounted, and then terminates.

6. The kernel then mounts the normal root filesystem.

7. Now that the filesystem is accessible and intact, the boot loader can be installed.

8. The boot loader is configured to load into /dev/initrd a filesystem with the set of modules that was used to bring up the system. (e.g., device /dev/ram0 can be modified, then unmounted, and fi? nally, the image is written from /dev/ram0 to a file.)

9. The system is now bootable and additional installation tasks can be performed.

The key role of /dev/initrd in the above is to reuse the configuration data during normal system operation without requiring initial kernel selection, a large generic kernel or, recompiling the kernel.

A second scenario is for installations where Linux runs on systems with different hardware configurations in a single administrative network.

In such cases, it may be desirable to use only a small set of kernels (ideally only one) and to keep the system-specific part of configura? tion information as small as possible. In this case, create a common file with all needed modules. Then, only the /linuxrc file or a file executed by /linuxrc would be different.

A third scenario is more convenient recovery disks. Because informa? tion like the location of the root filesystem partition is not needed at boot time, the system loaded from /dev/initrd can use a dialog and/or auto-detection followed by a possible sanity check.

Last but not least, Linux distributions on CD-ROM may use initrd for easy installation from the CD-ROM. The distribution can use LOADLIN to directly load /dev/initrd from CD-ROM without the need of any floppies. The distribution could also use a LILO boot floppy and then bootstrap a bigger RAM disk via /dev/initrd from the CD-ROM.

FILES

/dev/initrd

/dev/ram0

/linuxrc

/initrd

NOTES

1. With the current kernel, any filesystems that remain mounted when /dev/ram0 is moved from / to /initrd continue to be accessible. However, the /proc/mounts entries are not updated.

2. With the current kernel, if directory /initrd does not exist, then /dev/ram0 will not be fully unmounted if /dev/ram0 is used by any process or has any filesystem mounted on it. If /dev/ram0 is not fully unmounted, then /dev/ram0 will remain in memory.

3. Users of /dev/initrd should not depend on the behavior given in the above notes. The behavior may change in future versions of the Linux kernel.

SEE ALSO

chown(1), mknod(1), ram(4), freeramdisk(8), rdev(8)

Documentation/admin-guide/initrd.rst (or Documentation/initrd.txt be?

fore Linux 4.10) in the Linux kernel source tree, the LILO documenta?

tion, the LOADLIN documentation, the SYSLINUX documentation

COLOPHON

This page is part of release 5.10 of the Linux man-pages project.   A

description of the project, information about reporting bugs, and the

latest   version   of   this   page,   can   be   found   at

https://www.kernel.org/doc/man-pages/.

Linux   2019-03-06   INITRD(4)