

Linux tip: Finding rootfs during boot

Four tips on how to "rediscover" a missing root file system after adding storage devices and adapters

Skill Level: Introductory

[Lei Jiang \(jlei@cn.ibm.com\)](mailto:jlei@cn.ibm.com)

Staff Software Engineer

IBM

11 Mar 2009

As a Linux® administrator, you may encounter `rootfs` errors like `cannot mount rootfs` and `kernel panic` when you try to reboot a server after attaching volumes from external storage or even after installing a new Linux operating system.. This article outlines the Linux booting process on an x86 platform, shows why this problem happens, and offers four tips to avoid it or fix it.

Outlining the problem

The root file system (named `rootfs` in our sample error message) is the most basic component of Linux. A root file system contains everything needed to support a full Linux system. It contains all the applications, configurations, devices, data, and more. Without the root file system, your Linux system cannot run.

You may have experienced the `cannot mount rootfs` error (followed by your Linux host getting a `kernel panic`) after you reboot your system. This will most likely happen after you have attached some storage volumes from external storage. Or, it can also happen when you have finished copying files and need a reboot to finish installation.

Want more Linux tips?

- [Job scheduling with cron and at](#)

- [Bash parameters and parameter expansions](#)
- [Bash test and comparison functions](#)
- [Printing DVI files with CUPS](#)
- [Controlling the duration of scheduled jobs](#)
- [All Linux tips](#)

Don't see what you are looking for? [Let us know](#) and we will try to find it for you!

If this happens, your system will not restart. It could take time to troubleshoot this problem and fix it. This set of tips should help you solve the problem and save time.

Another culprit could be the fact that your Linux kernel needs to mount the root file system, but it can't find the target device. In other words, your root file system doesn't show up where it is supposed to be. For example, you install your Linux root file system on disk `/dev/sda`, but your system can't mount it during reboot. There are two possible reasons for this:

1. Disk `/dev/sda` is not showing up during your system reboot. This might occur because your Linux host missed loading the key driver for your root file system. This is unlikely. The Linux install program will build all drivers needed into the `initrd` image so the Linux system can easily load the device drivers during boot. However, if you've installed drivers manually, this error is possible.
2. Disk `/dev/sda` is showing up, but it's not your root file system. Your root file system has changed to `/dev/sdb` after reboot. This is most likely the case.

So how does `/dev/sda` get renamed as `/dev/sdb` after your system reboots? In Linux, `/dev/sd*` means *SCSI device*. Linux will name these devices from `sda` to `sdz` (then `sdaa` to `sdzz` and so on). It will name the first SCSI device `/dev/sda`, the second `/dev/sdb`, and so on.

If you install a device in an adapter with a driver that loads before your root file system device adapter driver (which is originally at `/dev/sda`), then your root file system gets moved one letter down the chain of command (`/dev/sdb`), and the root file system is not the first device encountered. Thus, it can't mount `rootfs` if it can't find it at the head of the line.

This is a straightforward explanation for what happens, but to add a little more context to the situation, let's outline the Linux boot process. (If you understand the "root" causes, feel free to skip directly to the [four tips on fixing it](#).)

How Linux booting works

Boot loaders

Linux Loader (LILO) is a generic boot loader for Linux that does not depend on a specific file system and can boot an operating system from floppy and hard disks. You can choose from up to sixteen different images at boot time, and you can set various parameters (like root device) independently for each kernel. LILO can be positioned in the master boot record (MBR) or the boot sector of a partition; if you do the second option, something else must be placed in the MBR to load LILO.

Another boot loader is GNU GRUB (GRUB). GRUB is the reference implementation of the Multiboot Specification, which allows a user to have several different operating systems on his or her computer at once and to choose which one to run. GRUB is dynamically configurable, loading its configuration at startup and allowing boot-time changes, such as selecting different kernels or initial RAM disks. GRUB is highly portable, supports multiple executable formats, and is geometry-translation independent. GRUB can download operating system images from a network, so it can support disk-free systems. GRUB supports automatic decompression of operating system images prior to booting from them. And unlike other boot loaders, it can communicate with a user directly from a GRUB prompt.

The following steps outline how the Linux boot process works:

1. The first thing a computer does on start-up is a primer test, POST (Power On Self Test). Several devices are tested, including the processor, memory, graphics card, and the keyboard. Also, the boot medium (hard disk, floppy unit, and CD-ROMs) is tested. After POST, the loader from a ROM loads the boot sector, which then loads the operating system from the active partition. You can change boot medium sequence by editing the server BIOS.
2. The boot sector is always at the same place—track 0, cylinder 0, head 0 of the boot device. This sector contains a program called `loader` (for Linux it is usually LILO or GRUB); this actually boots the operating system. Either loader is installed at the MBR or at the first sector of the active primary partition.
3. If you have multiple operating systems installed on your server, you need to select which one you want from the boot loader menu. You can also select which kernel to load in this menu if you have multiple kernels installed.
4. Then, the boot loader decompresses and loads the kernel. The kernel will

load kernel modules first, then detect hardware (floppy drive, hard disk, network adapters, etc.), verify hardware configuration, and then scan and load device drivers.

5. At this stage, the kernel will mount the root file system and system files. The location of system files is configurable during recompilation or with other programs. If the mount fails, a `kernel panic` will occur, and the system will freeze. This is the type of mount failure mentioned earlier.
6. Next, the kernel will start the system initialization process `init` which will become process number one. It will then start the rest of the system. The `init` process is Linux's first process, parent to all the other processes. This process is the first running process on any Linux/UNIX® system; it always has a PID of 1.
7. Then, the `init` examines the file `/etc/inittab` to determine what processes have to be launched. This file provides `init` information on runlevels and on which processes should be launched on each runlevel. After that, `init` looks up the first line with a `sysinit` (system initialization) action and executes the specified command file, like `/etc/rc.d/rc.sysinit` in Red Hat Linux. After the execution of the scripts in `/etc/rc.d/rc.sysinit`, `init` starts to launch the processes associated with the initial runlevel. At the end of execution of runlevel initial scripts, Linux will allow you to log in.

Each of the solutions discussed deals with step 5 from this list.

The four tips

Since the `cannot mount rootfs` error is mostly caused by device order, changing the device order or changing the driver loading sequence will solve this problem.

These two approaches can be implemented in the following four ways (each method is designed to let your Linux root disk show up to kernel/system in the first spot so that it can always use `/dev/sda` as device name):

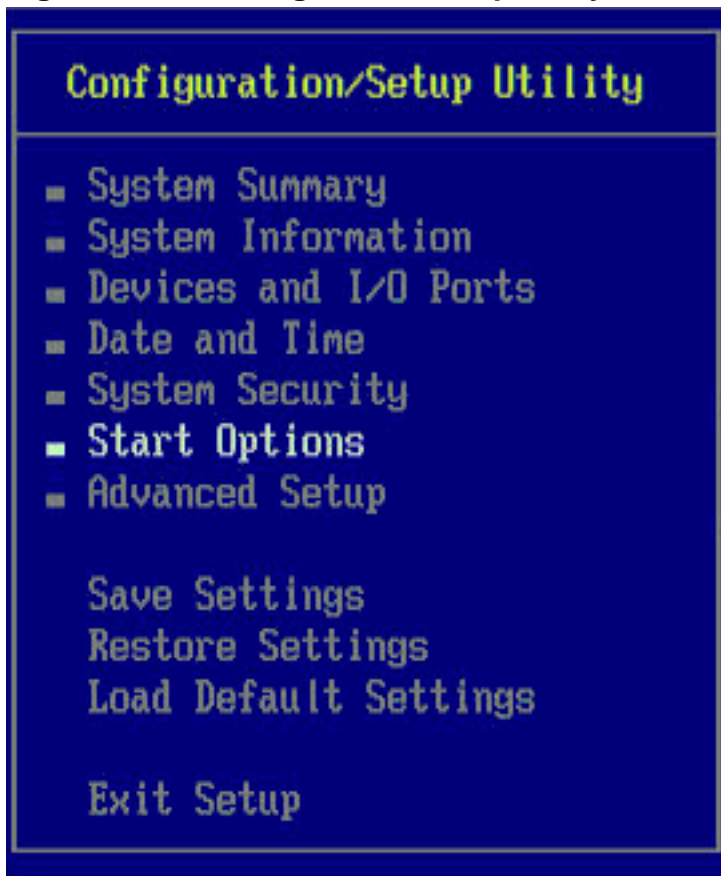
1. Change PCI device booting order in the server BIOS.
2. Plug your root file system disk into the first slot of its adapter.
3. Edit your `initrd` image to let the root file system adapter driver load earlier than all other storage adapter drivers.
4. Use the label, UUID, or friendly name instead of the device name for the root file system mount.

1. Change PCI device booting order in the server BIOS

If you are using IBM System x® or IBM BladeCenter® HS series, the procedure is:

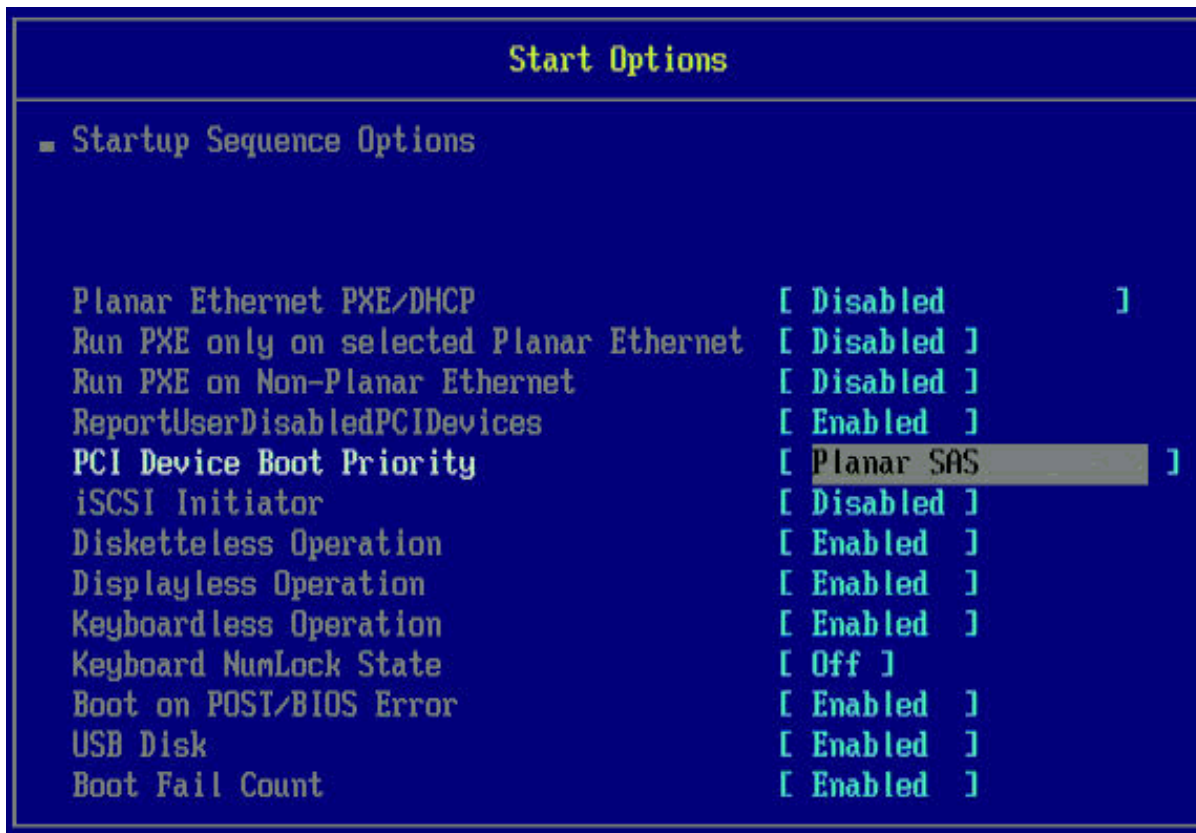
1. Press F1 when your server posts self-test info and prompts you to *Press F1 to enter BIOS*.
2. Select **Start Options** and press **Enter** as in Figure 1.

Figure 1. The configuration/setup utility



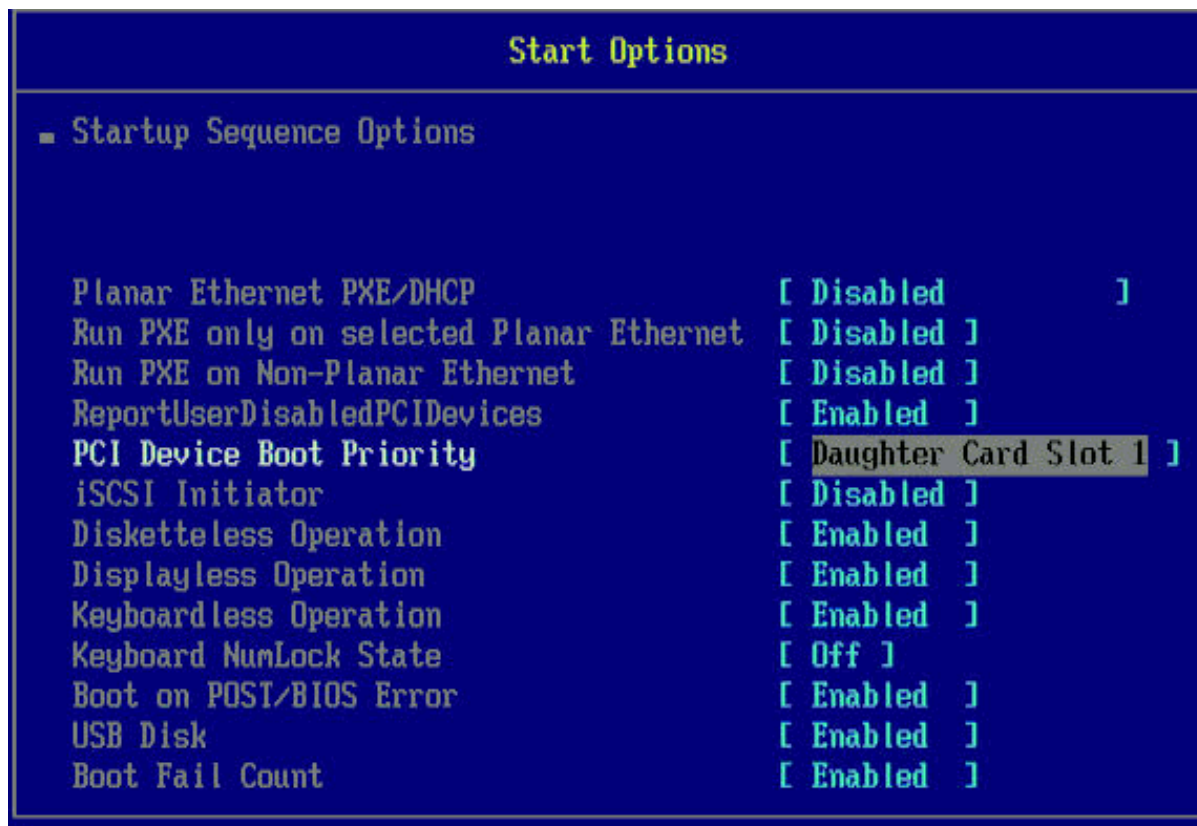
3. Change *PCI Device Boot Priority* to the adapter your root file system disk is using (as in Figure 2 below), if your Linux is installed on a local *Planar SAS* disk.

Figure 2. Start options



Or, you can select the related PCI slot to let **Daughter Card Slot 1** show up to system first with lowest device ID (see Figure 3).

Figure 3. Another set of start options



In this case, your local *Planar SAS* will have the larger device ID. If you installed Linux to the *Planar SAS* disk with device name `/dev/sda` and then attached one new disk device to *Daughter Card Slot 1*, this new disk will use the device name `/dev/sda`, and your root file system disk will be shifted to `/dev/sdb`. This will result in a `cannot mount rootfs` error.

2. Plug your root file system disk into the first slot of its adapter

If your root file system disk adapter can attach more than two devices, or you have more than two disk adapters, make sure your root file system disk is located in the slot with lowest device ID (such as the first slot of your first adapter). If there is a new added disk with a lower ID than your root file system disk, you need to put your root file system disk in a slot before it. This will make your root file system disk show up first to your Linux system.

3. Edit the `initrd` image to change the storage adapter sequence

The `initrd` image is actually a RAM disk that contains a small file system with basic configuration files, binaries, libraries, and drivers. In this small file system, there is a root file system that will load into memory during boot and there is an initial script to load system device drivers; it will remount the real root file system disk after finishing all devices drivers load. The `cannot mount rootfs` error always happens when your system tries to switch the `initrd` root file system to the real root file system disk

with the device name in this configuration. You need to change device driver load sequence in the driver load script of the initrd image to let your root file system disk show up with the right device name *before* the system tries to remount the actual root file system. To do this:

1. Use a Linux rescue CD to boot your system to rescue mode since you can't access the root file system.
2. Mount your root file system to a temporary directory like `sysroot` (some Linux rescue CDs will do this automatically for you). For example, your root file system disk is recognized as `/dev/sdc` when you're booting your system to rescue mode and your root file system is using partition 3.

```
[root@linuxhost ~]#mount /dev/sdc3 sysroot
```

3. Switch your rescue system root to your `sysroot` so that you can access the root file system of the problem server system.

```
[root@linuxhost ~]#chroot sysroot
```

4. Extract the initrd image.
 - a. For Red Hat Enterprise Linux prior to version 3 and SUSE Linux Enterprise Server prior to version 9:

```
[root@linuxhost ~]#cp /boot/initrd-x.x.x-x.ELsmp.img
./initrd.gz
[root@linuxhost ~]#mkdir temp
[root@linuxhost ~]#gunzip ./initrd.gz
[root@linuxhost ~]#mount -o loop -t ext2 initrd temp
```

This will extract your initrd image to a file `initrd`; then you mount this file with the `ext2` file system type to mount point `temp`. After this, you can see all files packaged into the initrd image.

- b. For Red Hat Enterprise Linux version 4 and later or SUSE Linux Enterprise Server version 10 and later:

```
[root@linuxhost ~]#cp /boot/initrd-*.img ./initrd.gz
[root@linuxhost ~]#mkdir temp
[root@linuxhost ~]#cd temp
[root@linuxhost ~]#gzip -dc ../initrd.gz | cpio -id
```

As in these newer Linux distributions, the initrd image is also compressed with `cpio`. These commands will extract whole file system with its directory structure under directory `temp`. You should

be able to find all files in the initrd image under directory temp now.

5. Now you will find file init for Red Hat Enterprise Linux and SUSE Linux Enterprise Server version 10, or file linuxrc for SUSE Linux Enterprise Server prior to version 9, under the temp directory. This file is a simple Linux shell script that contains all commands for loading device drivers to system memory. You can edit it by text or other editors.
6. Edit file init or linuxrc to let your root file system disk adapter appear earlier than all other storage adapters in the same category. In most cases, your root file system disk will be a SCSI device with prefix sd; you need put all other storage adapters like FC or SAS HBA drivers after the driver that the root file system disk depends on.
 - a. For example, if your root file system's physical disk is attached to a SAS adapter, then your init file may have a field like the following:

```
.....
echo "Loading mptbase.ko module"
insmod /lib/mptbase.ko
echo "Loading mptscsi.ko module"
insmod /lib/mptscsi.ko
echo "Loading mptspi.ko module"
insmod /lib/mptspi.ko
echo "Loading mptsas.ko module"
insmod /lib/mptsas.ko
echo "Loading mptscsih.ko module"
insmod /lib/mptscsih.ko
echo "Loading qla2xxx.ko module"
insmod /lib/qla2xxx.ko
.....
```

In this case, put all other SCSI adapters drivers (like the Qlogic HBA driver qla2xxx.ko) after the SAS adapter driver mptsas.ko.

- b. If you are using SUSE Linux Enterprise Server version 10, it looks like the following:

```
modprobe scsi_mod $params
modprobe sd_mod $params
params=
for p in $(cat /proc/cmdline) ; do
  case $p in
    aacraid.*)
      params="$params ${p#aacraid.}"
      ;;
  esac
done
echo "Loading aacraid"
modprobe aacraid $params
modprobe scsi_transport_fc $params
modprobe firmware_class $params
params=
for p in $(cat /proc/cmdline) ; do
  case $p in
    qla2xxx.*)
      params="$params ${p#qla2xxx.}"
  esac
done
```

```
;;
esac
done
echo "Loading qla2xxx"
modprobe qla2xxx $params
```

7. Verify that your edition is correct so you won't miss loading any needed drivers (SAS, for example, needs `mptbase.ko`, `mptscsi.ko`, `mptspi.ko`, `mptsas.ko`, and `mptscsih.ko` to load together so you can't let `qla2xxx.ko` load before `mptscsih.ko`). Also, you might want to check for typos.
8. At this time, you need to recompress the `initrd` file system to an image and replace the original one under the `/boot` directory.
 - a. For Red Hat Enterprise Linux prior to version 3 and SUSE Linux Enterprise Server prior to version 9:

```
[root@linuxhost ~]#umount temp
[root@linuxhost ~]#gzip initrd
[root@linuxhost ~]#cp initrd.gz
/boot/initrd-2.x.x-x.img
```

- b. For Red Hat Enterprise Linux version 4 and later or SUSE Linux Enterprise Server version 10 and later:

```
[root@linuxhost ~]#find ./ | cpio -H newc -o >
../initrd
[root@linuxhost ~]#gzip initrd
[root@linuxhost ~]#cp initrd.gz
/boot/initrd-2.6.x-x.img
```

9. Congratulations! You fixed the problem. Now reboot and check the result.

4. Use the label, UUID, or friendly name

UUID

UUID means Universally Unique Identifier. It is an identifier standard used in software construction, standardized by the Open Software Foundation as part of the Distributed Computing Environment (DCE). UUIDs are designed to enable distributed systems to uniquely identify information without significant central coordination; information labeled with UUIDs can be combined into a single database without needing to resolve name conflicts. Significant uses of this standard include the Linux `ext2/ext3` filesystem, LUKS encrypted partitions, GNOME, KDE, Mac OS X, and the Microsoft® Globally Unique Identifiers.

Some Linux file system types (like `ext2`, `ext3`, `reiserfs`, `swap`, and `XFS`) support mounting the file system with a label instead of a device name, and you can use

UUID instead if your Linux system supports it. Also, you can use friendly name if your device driver supports it.

These methods need your Linux system to support these features (like Red Hat Enterprise Linux V4 and later or SUSE Linux Enterprise Server V9 and later). Because label, UUID, and friendly name will bond with a dedicated device forever, no matter what device ID it has or what device name it gets, your system will always find your root file system disk.

Use a label

1. Create a label when you create a file system, like the root, swap, or other file systems.

```
[root@linuxhost ~]#mkfs.ext3 -L ROOT /dev/sda1
[root@linuxhost ~]#mkfs.reiserfs -l OSROOT /dev/sdb2
[root@linuxhost ~]#mkfs.xfs -L XFSROOT /dev/sde3
[root@linuxhost ~]#mkswap -L SWAP0 /dev/sdb5
```

2. Add a label to your file system after it's created.

```
[root@linuxhost ~]#e2label /dev/sda1 PRIMARY_ROOT
[root@linuxhost ~]#reiserfstune -l OSROOT /dev/sdb2
[root@linuxhost ~]#xfs_admin -L DATA1 /dev/sdf4
```

3. Use a label in your system.

- a. Edit `/etc/fstab` to your Linux system so that your system will use a label to mount your file system instead of a device name. The following is a simple example of the content of `/etc/fstab`:

```
LABEL=ROOT          /          ext3    defaults
1 1
LABEL=BOOT          /boot      ext3    defaults
1 2
LABEL=SWAP          swap       swap    defaults
0 0
LABEL=HOME          /home     ext3    nosuid,auto
1 2
```

- b. Edit `grub.conf` of your Linux boot loader.

```
title Linux
  root (hd0,0)
  kernel (hd0,0)/vmlinuz ro root=LABEL=ROOT rhgb quiet
  initrd (hd0,0)/initrd-2.x.x-xx.img
```

Use UUID

1. Get UUID for your root device. Let's assume your root file system is located in disk device /dev/sda as in the example below:

```
[root@linuxhost ~]#scsi_id -g -s -u /block/sda
```

2. Using the following command to check the ID for your root file system device with the device UUID you get from step 1.

```
[root@linuxhost ~]#ls /dev/disk/by-id/<your device UUID>
```

3. Use UUID in your system by editing /etc/fstab as in this example:

```
/dev/disk/by-id/scsi-<your uuid>-part2 / ext3
defaults 1 1
```

Use a friendly device name

If you use the device mapper multi-path (DMMP) tool for your multi-path storage, then you can use a friendly name for your root file system to avoid the device being renumbered by DMMP after reboot.

1. Get UUID for your root device. Let's assume your root file system is located in disk device /dev/sda.

```
[root@linuxhost ~]#scsi_id -g -s -u /block/sda
```

2. Edit /etc/multipath.conf to add a friendly name for your root file system device as in the following:

```
multipaths {
    multipath {
        wwid <your disk UUID get from above>
    }
    command>
        alias OSROOT
}
}
```

3. Then, after you reboot your system or reload DMMP, you will have your root file system device with a device name like /dev/mapper/OSROOT (if your root file system is using partition 3 of this disk, then you will may

have a device name `/dev/mapper/OSROOT-part3` or `/dev/mapper/OSROOTp3`).

4. Edit `/etc/fstab` to use this friendly name for this root file system device as follows:

```
/dev/mapper/OSROOT-part3    /      ext3  defaults    1
1
```

5. Edit `/etc/grub.conf` to use this name when system boots up.

```
title Linux
    root (hd0,0)
    kernel (hd0,0)/vmlinuz ro
    root=/dev/mapper/OSROOT-part3 rhgb quiet
    initrd (hd0,0)/initrd-2.x.x-xx.img
```

6. At this point, you've done. You can reboot and check to see whether it works.

Conclusion

This article demonstrates how to set up your system to avoid or fix Linux system cannot mount `rootfs` errors and also provides background on the boot process of a Linux system.

Resources

Learn

- "[Applying mount namespaces](#)" (developerWorks, September 2007) gives a more advanced look at mounting the root file system and mount propagation.
- In the [developerWorks Linux zone](#), find more resources for Linux developers (including developers who are [new to Linux](#)), and scan our [most popular articles and tutorials](#).
- See all [Linux tips](#) and [Linux tutorials](#) on developerWorks.

Get products and technologies

- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- [Participate in the discussion forum for this content.](#)

About the author

Lei Jiang

Lei Jiang works at IBM China Systems and Technology Lab in Shanghai as part of the Open Systems Interoperability Validation Lab storage testing team. He has three years of SAN storage heterogeneous support and testing experience; his current focus is on storage technologies and open system SAN and high availability solutions.

Trademarks

IBM, the IBM logo, ibm.com, DB2, developerWorks, Lotus, Rational, Tivoli, and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. See the current list of [IBM trademarks](#).

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft is a trademark of Microsoft Corporation in the United States, other

countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.