Linux Device Naming

Like other operating systems, the Linux® OS must discover, enumerate, and assign default names to hardware devices—tape drives, network adapters, USB devices, and the like—before these devices can be used. This article discusses methods for Linux device enumeration, how enumeration relates to default assigned names, and how administrators can modify these names.

BY MATT DOMSCH AND AHMAD ALI

Related Categories:

Linux

Visit www.dell.com/powersolutions for the complete category index.

Names provide context for objects and create explicit or implied relationships between objects. For computers, however, an object name may have little to do with its underlying characteristics.

Consider a real-life anecdote of a child born at a hospital in Austin, Texas. The hospital gave the baby the name "BGC Dom." This name appeared on her wristbands and bassinet cart, the nurse's duty station whiteboard, and hospital paperwork. The hospital had created a naming convention designed to be easily recognizable to the staff, unambiguous (given an average of fewer than 20 babies at the hospital at a time), and protective of the mother's and baby's privacy: "BG" stood for "baby girl," "C" was the first letter of the mother's first name, and "Dom" was the first three letters of the mother's last name. (The convention also included extensions for identical twins.) Because her parents did not want to call her "BGC Dom" throughout her life, they filled out other paperwork to officially name her (involving the baby's globally unique footprints to identify her). However, even after the parents completed the official naming, the hospital continued to use the name they had assigned, "BGC Dom"—and yet no one was confused about who the baby was or who the mother was. This is the goal of any good naming system.

System device naming in the Linux OS is similar in several ways. Conventions are in place to assign default names to devices, but the default name may not be the only name a device has, and the device name can be changed to help ease identification and management of that device.

Understanding enumeration and naming

Enumeration is the process of moving through the system in some order: depth first, breadth first, by some list of subsystems, and so on. Each control entity (Linux kernel, BIOS, and system administrator) performs its own device enumeration and naming. Each discovered device is assigned a default name—for example, in Linux, Ethernet devices become eth0, eth1, eth2, and so on, and SCSI and Serial ATA (SATA) disks become /dev/sda, /dev/sdb, and so on. The Linux default naming convention is simply the device type (eth or sd) plus an incrementing value based on the order in which devices are found. The BIOS naming is "interrupt 13h device 80h, device 81h," and so on, while the system administrator may name devices as "my boot disk," "my home directories," "my database," and so on.

Such default naming conventions can present several problems:

- The default device name assigned by the OS may bear no resemblance to the name assigned by the system BIOS or hardware manufacturer documentation, and it may have no relationship to eventual system usage.
- A given device name may change if the system topology or enumeration order changes.
- As one device name changes, another device may take the default name previously assigned to the first device (/dev/sdb may become /dev/sda, eth1 may become eth0, and so on).
- Administrators are accustomed to thinking that the default name is the correct name or the only name. If the name changes, the administrator will have an inaccurate conception of the system or device.
- Tools and configuration files that use default device names may behave incorrectly if the names change.

In an ideal world, each device would have some characteristic besides enumeration order to use for naming. Just as in the scenario described earlier in which the baby's unique footprints are registered for her birth certificate, many (but not all) devices have some form of unique identifier: for example, Ethernet devices have 48-bit globally unique Media Access Control (MAC) addresses, and disks have manufacturer serial numbers or other globally unique identifiers stored in their firmware. The naming system could have tables and algorithms for assigning consistent names to devices independent of their enumeration order.

In an ideal world, each device would have some characteristic besides enumeration order to use for naming.

Linux 2.6 kernels and the udev tool enable administrators to assign arbitrary names to devices following their own naming conventions. The Linux device enumeration and default naming process varies depending on the type of device, with conventions for Ethernet devices differing from those for disk devices.

Naming Ethernet devices

Default naming for Ethernet devices uses device enumeration order; enumeration order is based on the order in which network device driver modules are loaded, which in turn is based on the order in which those drivers are listed in /etc/modprobe.conf. If a system has one Ethernet device that uses the Intel® e1000 driver and another that uses the Broadcom tg3 driver, administrators can affect the enumeration order of these devices based on the order of entries in /etc/modprobe.conf.

Multiple Ethernet devices using the same driver can be more complicated than single devices for each driver. If the drivers in the preceding example are being used by multiple devices, the e1000 driver enumerates all the Ethernet ports it recognizes (for example, eth0, eth1, and so on), and then the tg3 driver, when loaded, enumerates those it recognizes (eth2, eth3, and so on). Discovery order is based on the system-wide PCI device list (which itself was enumerated in a particular order) and the list of PCI device IDs that the driver can recognize. This enumeration makes no distinction between LAN on Motherboards (LOMs) and add-in PCI cards, which can cause two LOMs to be enumerated backward compared to their BIOS names (NIC1 as eth1, NIC2 as eth0), and cause add-in cards to be assigned eth numbers that are lower than the LOM port names, leading to confusion for system administrators.

Because each Ethernet port has a unique MAC address, Linux can assign any name to a given MAC address. Administrators have traditionally used the `nameif` command and the related /etc/mactab file to assign logical names (public0, private1, subnet15, and so on) to each interface. Red Hat® Enterprise Linux 3 and 4, however, have configuration files in /etc/sysconfig/network-scripts/ifcfg-*devicename* that include a `HWADDR=`*`xx:xx:xx:xx:xx:xx`* value used to assign a device name to the Ethernet port based on the specified MAC address, overriding the default name assigned during device enumeration.

Novell® SUSE® Linux Enterprise Server (SLES) 9 and 10 both store configurations in files like /etc/sysconfig/network/ifcfg-eth-id-*xx*:*xx*:*xx*:*xx*:*xx*:*xx*, where the last part of the file name is the Ethernet port's MAC address. By default, the OS assigns eth0 to the first device found. In SLES 10, the udev tool maintains name persistence for Ethernet ports by renaming ports from the kernel default naming scheme. This tool establishes naming rules using files in /etc/udev/rules.d; it also has rules for adding Ethernet devices. The /lib/udev/rename_netiface script carries out this renaming. Administrators can choose to establish new rules or modify default ones. Each rule is a single line with keywords used to match or assign a value or perform an action. The "Using the udev tool for device naming" sidebar in this article provides some example rules.

Dell engineers have developed a script that assigns Ethernet device names: the LOMs are assigned names that match BIOS and documentation-defined order, and Ethernet ports on add-in cards are then named in increasing slot number order (the slot numbers are

USING THE UDEV TOOL FOR DEVICE NAMING

The udev tool provides device file management capabilities in Linux distributions based on the 2.6 kernel. It uses rules to dynamically maintain (create, remove, or rename) files for devices that are actually present in the system. It is based on sysfs and hotplug, and replaces devfs and static device node naming.

The default location for udev configuration files is /etc/udev. The main configuration file is /etc/udev/udev.conf. This file has information such as udev_root, udev_rules, permissions, and logging priority. The default udev_root location is /dev; the default udev_rules location is /etc/udev/rules.d.

The udev rules are stored in files under the udev_rules directory and are processed in lexical order. Every non-blank or non-comment line in a rule file is a rule. These rules contain one or more key value pairs separated by a comma. These are both match keys and assignment keys. The rules are applied when all match keys are matched; when no rules match, default device names are used.

Rules may name a device, add a symlink, or run a program to take other actions like renaming an Ethernet port. The key match operators are = and !=; the assignment operators are =, :=, and +=. Key value assignments can be changed by subsequent rules unless the final assignment operator, :=, is used.

Figures A–E show some example rules. *Note:* The exact syntax for udev rules has changed over time; for an OS distribution's specific udev syntax, see the distribution documentation.

```
SUBSYSTEM=="net", ACTION=="add", SYSFS{address}=="xx:xx:xx:xx:xx:xx",
    IMPORT="/lib/udev/rename_netiface %k eth0"
```

Figure A. Example udev rule that renames Ethernet ports at a given MAC address to eth0 with the help of an external script

```
KERNEL=="*[!0-9]", IMPORT{program}="/sbin/edd_id --export $tempnode"
KERNEL=="*[!0-9]", ENV{ID_EDD}=="?*",
    SYMLINK+="disk/by-id/edd-$env{ID_EDD}"
KERNEL=="*[0-9]", ENV{ID_EDD}=="?*",
    SYMLINK+="disk/by-id/edd-$env{ID_EDD}-part%n"
```

Figure B. Example udev rule that creates a symlink for disks and partitions using BIOS disk data, providing persistent names based on BIOS disk enumeration

```
BUS="scsi", PROGRAM="/sbin/udev.get_persistent_device_name.sh",
    NAME="%k" SYMLINK="%c{1+}"
```

Figure C. Example udev rule that adds a symlink with a helper program that provides path and UUID information

```
KERNEL=="sd*[!0-9]|sr*|st*", ENV{ID_SERIAL}=="",
    IMPORT{program}="/sbin/scsi_id -g -x -s %p -d $tempnode"
KERNEL=="sd*[!0-9]|sr*|dasd*[!0-9]", ENV{ID_SERIAL}=="?*",
    SYMLINK+="disk/by-id/$env{ID_BUS}-$env{ID_SERIAL}"
KERNEL=="sd*[0-9]|dasd*[0-9]", ENV{ID_SERIAL}=="?*",
    SYMLINK+="disk/by-id/$env{ID_BUS}-$env{ID_SERIAL}-part%n"
```

Figure D. Example udev rule using the SLES 10 syntax that adds symlinks for disks and all partitions

```
BUS=="scsi", ENV{ID_SERIAL}=="360045600007890012", NAME="BackupDisk4%n"
KERNEL=="sd*[!0-9]", ENV{ID_SERIAL}=="360045600007890012",
    SYMLINK+="location/Rack1Shelf2Disk4"
```

Figure E. Example udev rules that provide a persistent friendly name and symlink based on SCSI disk serial number printed inside the system). This script is available at linux.dell.com/files/name_eths, and works on Red Hat Enterprise Linux and SLES 10.

Naming disk devices

Default naming for disk devices is also based on device enumeration order, which in turn is based on the order in which the device drivers are loaded. For standard IDE disks and CD/DVD drives, the OS discovers these drivers during kernel startup, before other kernel modules are loaded. IDE devices are named /dev/hda, /dev/hdb, and so on. Following the monolithic kernel startup, enumeration continues by loading the drivers listed in the initial ramdisk, which is generated from the order listed in /etc/modprobe.conf. SCSI, SATA, Serial Attached SCSI (SAS), Fibre Channel, and the various hardware RAID controller drivers are all modular, and thus are loaded from the initial ramdisk. Because these drivers are based on the SCSI midlayer, the OS names the devices /dev/sda, /dev/sdb, and so on.

Administrators should note that the Linux kernel discovery order has no relationship with the BIOS boot order—the disk that Linux calls /dev/hda or /dev/sda may not be the disk the BIOS uses to boot the OS. This is usually only a problem during OS installation, because administrators must ensure that the boot loader, /boot partition, and often the root (/) partition are on the disk the BIOS uses to boot the OS. Further complicating the problem is that Linux may not be able to ask the BIOS directly for its list of devices.

Dell engineers have developed a method to maintain device naming consistency for disks. By writing a system-wide unique signature into the master boot record (the first sector) of each disk in the system, the installation software can use this identifier to match the BIOS device list with the Linux device list. Early in the kernel startup process, each disk is read, and the signature value is stored in memory and later exposed through sysfs. Linux user-space tools such as the Red Hat Enterprise Linux 4 Anaconda installer and SLES 10 YaST (Yet Another Setup Tool) installer can then read the signatures from all the disks, compare them to the BIOS disk signatures shown in sysfs, and map the BIOS device names to the Linux device names.

After the system is fully installed, the standard method to mount the file systems in the proper place is to use file system labels in the /etc/fstab file. For example:

```
LABEL=/         /         ext3    defaults 1 2
LABEL=/boot     /boot     ext3    defaults 1 2
```

This example indicates that the ext3 file system with the / file system label should be mounted at the mount point /, and the /boot file system label should be mounted at the mount point /boot. As long as the file system labels in use are unique across the system, the file systems can then be mounted in the correct location. In Red Hat Enterprise Linux 3 and later, the boot loader configuration file can use this label to find the root file system. Administrators should pass `root=LABEL=/` on the kernel command line, although they must use an initial ramdisk for this to work, because the shell in the initial ramdisk resolves the requested label to the actual underlying disk partition and file system.

Using label-less file systems

Some file systems (such as, until recently, swap devices) and raw devices do not include useful file system labels, but administrators can use other methods to assign consistent names to these devices. For Red Hat Enterprise Linux 3, Dell has developed the devlabel tool, which uses unique SCSI serial numbers or universally unique identifiers (UUIDs), depending on the individual disk capability, and a configuration file to map the identifiers to the desired mount point. For Red Hat Enterprise Linux 4, SLES 9, and SLES 10, this capability has been superseded by udev. Administrators can set the udev rules to provide persistent symbolic links (symlinks) using UUID, path, serial number, or file system labels. For example udev rules, see the "Using the udev tool for device naming" sidebar in this article.

Divorcing enumeration from naming enables administrators to assign device names based on local logical usage and to change the physical topology of systems without affecting administrator expectations or system software.

Controlling Linux device naming

Although Linux assigns default names to all types of devices, these names may not be the most useful names. Fortunately, administrators can use a number of methods to change the names depending on their needs. Divorcing enumeration from naming enables administrators to assign device names based on local logical usage and to change the physical topology of systems without affecting administrator expectations or system software.

Matt Domsch is a Linux software architect at Dell and a member of the Fedora Project Board. Matt has a B.S. in Computer Science and Engineering from the Massachusetts Institute of Technology and an M.S. in Computer Science from Vanderbilt University.

Ahmad Ali is a member of the Dell Linux Engineering team. He has a B.S. in Electrical Engineering and an M.S. in Computer Engineering from Florida Atlantic University.