

Resolving Device Renaming Issues in Linux

Both the addition of storage devices to an environment running the Linux® operating system (OS) and storage failures within the Linux OS can cause storage device name inconsistencies between reboots. Dell has developed software called *devlabel* that helps to resolve storage device renaming issues within the Linux OS. The *devlabel* software establishes a symbolic link layer between the user and the storage, controlling and keeping consistent disk accessibility. Furthermore, using *devlabel* within a storage area network (SAN) can considerably ease SAN deployment in a shared storage environment.

BY GARY LERHAUPT

Previously, for Oracle® database deployments utilizing raw devices, administrators could avoid storage device renaming by using a Dell™ service named *saferawdevices*. This service employed the inherent unique device identifier of a storage device to tie the device to the raw devices used by Oracle software.¹ However, this solution repaired only part of the problem; device renaming affects not only raw devices, but SCSI and Integrated Device Electronics (IDE) hard drives as well.

The Linux® operating system (OS) lists storage devices in the `/dev` directory. Linux prefaces SCSI disks with “sd” and IDE disks with “hd.” For example, the third partition on the first SCSI disk would be called `/dev/sda3`, while the third partition on the first IDE disk would be called

`/dev/hda3`. Moreover, Linux does not automatically write signatures on its disks to keep track of them. Given two SCSI disks `/dev/sdb` and `/dev/sdc`, if the disk associated with `/dev/sdb` were to fail, on the following boot the original `/dev/sdc` disk would likely be renamed as `/dev/sdb` (see Figure 1). This renaming complicates users’ ability to access their data. The *devlabel* software from Dell includes the functionality of the *saferawdevices* service and extends its methodology to protect against all instances of storage device renaming within Linux.

Using symbolic links to keep device names consistent

To continue the work developed to assist Oracle deployments, Dell has proposed a solution for the entire device

¹For more details on Dell efforts to resolve storage device renaming problems in Oracle deployments, see “Safeguarding Oracle Databases Deployed on Linux-based Platforms” by Gary Lerhaupt in *Dell Power Solutions*, November 2002.

renaming issue. Called *devlabel*, the software creates a symbolic link (symlink) layer between the user and the storage device. The user references all storage devices by the symlink that, in turn, points to the internal storage device name. If the underlying storage device name changes, *devlabel* reconfigures the symlink to point to the new name of the storage device. In this way, users can simply refer to their storage through a consistent and unchanging symlink name.

To set up *devlabel*, administrators simply determine what convention to use for naming their symlinks and then begin adding symlinks to their storage devices using the `devlabel add` command. During the procedure, *devlabel* will probe all storage devices found on a system to confirm that the identifier returned by the device in question is indeed unique. If *devlabel* cannot find a unique identifier, *devlabel* will refuse to add the symlink for the device.

From this point forward, during boot, the `rc.sysinit` script will call `devlabel restart` to readjust symlinks if necessary. The administrator can also manually call `devlabel restart` at any time. To later remove symlinks, administrators can use the `devlabel remove` command.

Although all configuration settings are kept in the file `/etc/sysconfig/devlabel`, administrators should not edit this file by hand. Instead, `devlabel status` will give pertinent information on the current condition of all symlinks.

Comparing devlabel to other labeling infrastructures

Most Linux file systems already have built-in labeling infrastructures that provide functionality similar to that of *devlabel*. For example, administrators can use the *e2label* program with `ext2` or `ext3` file systems to write a unique label within the file system. Afterward, these partitions can be mounted by label, thus assuring consistent data access. However, the file system labeling method is far from ubiquitous. Disparate file systems each

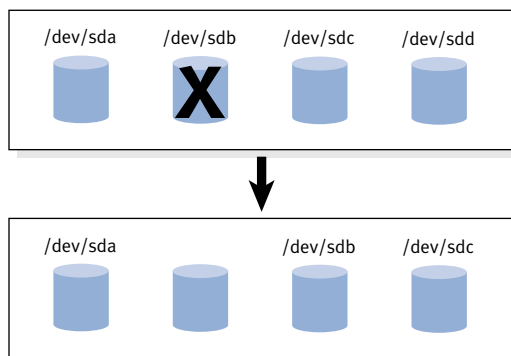


Figure 1. Possible disk renaming after rebooting under the Linux operating system

To set up *devlabel*, administrators simply determine what convention to use for naming their symlinks and then begin adding symlinks to their storage devices using the `devlabel add` command.

maintain their own implementation for handling the labeling process, and certain partitions, such as swap partitions, may have no file system at all.

Compared to *e2label*, the *devlabel* method appears to be more convenient and elegant. Moreover, with its integration into the hot-plug system, it also allows for consistent access to hot-pluggable storage. Using *devlabel*, administrators can hot plug a USB, IEEE® 1394 (FireWire), or PCMCIA (Personal Computer Memory Card International Association) storage device, add a symlink to it, and then consistently access data through the symlink. Once the device is disconnected from the system, *devlabel* removes the symlink, but as soon as the device is hot plugged again, *devlabel* will automatically re-create the symlink to allow access. Although Linux does not yet implement SCSI hot plug, once this functionality is added, *devlabel* will be a ready and apt structure for accessing any hot-plugged SCSI drives.

Using devlabel in SAN environments

The *devlabel* software can also help to reduce the time and effort needed for storage area network (SAN) deployment in a shared storage environment. The *devlabel* program uses a unique identifier to map symlinks to disk drives. Because the disks seen by the Linux host in this scenario are the same for all Linux hosts within the SAN, all the hosts see the same unique identifiers. Consequently, administrators can copy the *devlabel* configuration file from any one node to any other node in the SAN, and *devlabel* will automatically configure that node accordingly. Thus, once *devlabel* is installed throughout the nodes on the SAN, an administrator can configure the entire SAN deployment simply by copying a single *devlabel* configuration file to each node.

Looking further into this method, configuring the storage for nodes in a SAN requires only a few steps when using *devlabel* (see Figure 2). Once administrators create and partition the proper storage configuration on the storage hardware, and once they place all of the appropriate fiber connectivity between the SAN nodes, administrators should install *devlabel* on an initial master node. For each partition shared in the SAN, a proper symlink binding is created using the `devlabel add` command: for example, `devlabel add /dev/disk1part1 /dev/sdb1`. Then mount

points are created: for example, with the command `mkdir /mnt/disk1part1`. In the final step, an entry is added into `/etc/fstab` to automatically mount these partitions to their specified mount points: for example, `/dev/disk1part1 /mnt/disk1part1 auto defaults 0 0`. Once accomplished, this step completes the master node setup.

Setting up additional nodes involves copying files from the master node. Administrators first install `devlabel` on the secondary node. They then copy the master node's `devlabel` configuration file, `/etc/sysconfig/devlabel`, to the secondary node along with the relevant portions of the master node's `/etc/fstab` file. Identical mount point directories are then created. Execution of the `devlabel restart` command completes the deployment. Again, because `devlabel` relies on unique disk identifiers, even if the secondary node sees a disk as `/dev/sdf` and the master node sees it as `/dev/sdb`, `devlabel` will sort out the discrepancy and point the symlink to the proper location on each node.

Not only does `devlabel` significantly decrease the effort behind SAN deployment, it also ensures that exactly the same data can be accessed from exactly the same mount point on every node within the SAN.

Using devlabel in cluster environments

As in a SAN deployment, `devlabel` can be used to deploy shared raw device access in a clustered shared storage environment. Since `devlabel` treats raw devices as a special type of symlink within its scripts, the same consistent access carries over to raw devices. Administrators can configure a master node with all the appropriate raw device to storage device mappings and then transfer the `/etc/sysconfig/devlabel` configuration file of the master node to all other nodes within the cluster. Again, because all other nodes within the cluster see the same storage, they are automatically able to create the proper raw mappings to the specified storage based on the captured universal unique identifiers (UUIDs) in the configuration file. This process can greatly simplify deploying an Oracle9i[™] Real Application Clusters configuration.

Increasing devlabel usability

The `devlabel` software offers a powerful storage management

Not only does `devlabel` significantly decrease the effort behind SAN deployment, it also ensures that exactly the same data can be accessed from exactly the same mount point on every node within the SAN.

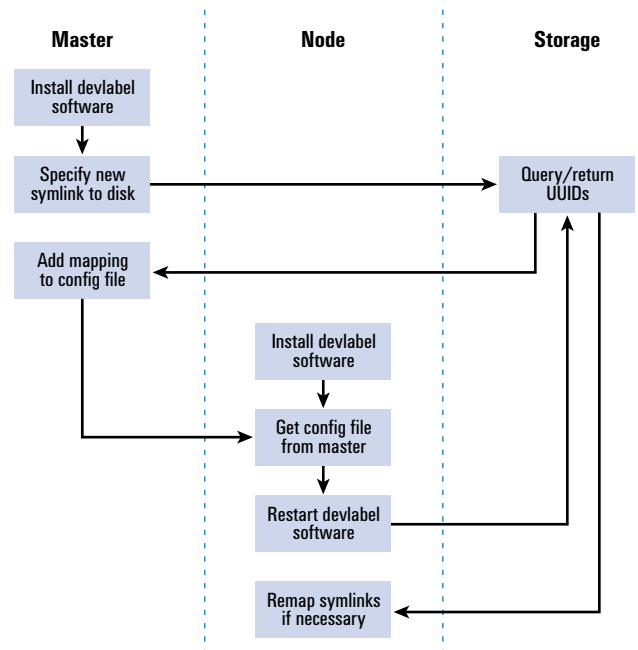



Figure 2. Method for configuring nodes in a SAN deployment using `devlabel`

tool in a small package. Initially developed to ensure consistent data access that circumvents Linux device renaming issues, `devlabel` has quickly become adaptable to hot-pluggable storage devices, SAN deployments, and Oracle9i Real Application Clusters deployments.

For future Linux kernels, plans are underway for extending `devlabel` to use identifiers as provided by the `driverfs` virtual file system, which will increase `devlabel` usability and the likelihood that unique identifiers can be found for all types of storage devices. As development continues on `devlabel`, the latest version can be downloaded from <http://www.domsch.com/linux/devlabel>. 

Gary Lerhaupt (gary_lerhaupt@dell.com) is a software engineer in the Linux Development Team of the Dell Product Group. He also collaborates on the Dell Oracle9i Real Application Clusters (RAC) initiative deployed on Red Hat[®] Linux. Gary has a B.S. in Computer Science and Engineering from The Ohio State University and is also a Red Hat Certified Engineer (RHCE).

FOR MORE INFORMATION

Dell and Linux: <http://www.dell.com/linux>
`devlabel`: <http://www.domsch.com/linux/devlabel>