Global Knowledge™

Written and Provided by

Expert Reference Series of White Papers

10 Red Hat® Linux™ Tips and Tricks

10 Red Hat® Linux™ Tips and Tricks

Compiled by Red Hat Certified Engineers

Introduction

Are you looking for a quick and simple reference guide to help you navigate Red Hat® Linux™ systems?

Look no further! Global Knowledge and Red Hat have assembled these 10 Tips and Tricks from Red Hat Certified Engineers® (RHCEs) to give you an edge on managing these systems.

1. Wiping a Hard Drive

By Dominic Duval, Red Hat Certified Engineer

Have you ever needed to completely wipe out critical data from a hard drive? As we all know, **`mkfs`** doesn't erase a lot. (You already knew this, right?) **`mkfs`** and its variants (e.g., **`mkfs.ext3`** and **`mke2fs`**) only get rid of a few important data structures on the filesystem, but the data is still there! For a SCSI disk connected as **`/dev/sdb`**, a quick

```
dd if=/dev/sdb | strings
```

will let anyone recover text data from a supposedly erased hard drive. Binary data is more complicated to retrieve, but the same basic principle applies: the data was not completely erased.

To make things harder for the bad guys, an old trick was to use the 'dd' command as a way to erase a drive.

Note: This command *will* erase your disk!

```
dd if=/dev/zero of=/dev/sdb
```

There's one problem with this: newer, more advanced, techniques make it possible to retrieve data that were replaced with a bunch of 0s. To make it more difficult, if not impossible, for the bad guys to read data that was previously stored on a disk, Red Hat ships the "shred" utility as part of the coreutils RPM package. Launching "shred" on a disk or a partition will write repeatedly (25 times by default) to all locations on the disk.

Note: Be careful with this one too!

```
shred /dev/sdb
```

This is currently known to be a very safe way to delete data from a hard drive before, let's say, you ship it back to the manufacturer for repair or before you sell it on eBay!

2. How To Determine the Manufacturer of a Laptop Battery

By Dominic Duval, Red Hat Certfied Engineer

With all the recent news about laptop batteries suddenly exploding, it might be a good idea to determine the manufacturer and model number of the battery that's currently connected to your laptop.

A simple file, included with the 2.6 kernel that runs on Red Hat Enterprise Linux 4, can easily show this information on any laptop running with ACPI enabled:

```
cat /proc/acpi/battery/BAT0/info
```

Look for the "model number" and "OEM info" fields.

3. Sharing a Hot Spare Device in Software RAID

By Forrest Taylor, Red Hat Certified Engineer

Have you ever wondered if you could share a hot spare device between two software RAID arrays? You can share a hot spare device if you put mdadm in daemon mode and have it poll your RAID arrays.

Let's assume that you have two RAID 1 arrays with one hot spare configured in this manner:

```
/dev/md0 RAID1
--
/dev/sda1
/dev/sdb1

/dev/md1 RAID1
--
/dev/sdc1
/dev/sdd1
/dev/sde1 (Hot Spare)
```

This setup shows **/dev/md0** with two devices, and **/dev/md1** with three devices, with **/dev/sde1** as a hot spare. In this scenario, you want to share **/dev/sde1** with **/dev/md0** if it should need it. To do that, you must configure the **/etc/mdadm.conf** file and define a spare-group name.

In **/etc/mdadm.conf**, start off by listing all of the devices:

```
echo "DEVICE /dev/sda1 /dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1"
>> /etc/mdadm.conf
```

Scan the RAID arrays for the current details, and add it to the file:

```
mdadm -D -s >> /etc/mdadm.conf
```

/etc/mdadm.conf should now contain something like the following:

```
# Caution, the ARRAY and UUID should be on the same line.
```

```
DEVICE /dev/sda1 /dev/sdb1 /dev/sdc1 /dev/sdd1
/dev/sde1
ARRAY /dev/md0 level=raid1 num-devices=2
UUID=29bc861f:6f1c72b0:162f7a88:1db03ffe
devices=/dev/sda1,/dev/sdb1
ARRAY /dev/md1 level=raid1 num-devices=2
UUID=aee2ae4c:ec7e4bab:51aefe40:9b54af78
devices=/dev/sdc1,/dev/sdd1,/dev/sde1
```

At this point, you need to create a spare-group entry for each array. The name does not matter, as long as it is the same for each array that you want to share the hot spare device(s).

Here, we choose "shared" as the name of the spare-group and add an entry for each ARRAY in the **/etc/mdadm.conf** file:

```
# Caution, the ARRAY and UUID should be on the same line.

DEVICE /dev/sda1 /dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1
ARRAY /dev/md0 level=raid1 num-devices=2
UUID=29bc861f:6f1c72b0:162f7a88:1db03ffe
devices=/dev/sda1,/dev/sdb1
spare-group=shared
ARRAY /dev/md1 level=raid1 num-devices=2
UUID=aee2ae4c:ec7e4bab:51aefe40:9b54af78
devices=/dev/sdc1,/dev/sdd1,/dev/sde1
spare-group=shared
```

Once the configuration file is ready, **mdadm** can run in daemon mode and poll the devices. If **mdadm** determines that a device has failed, it will look for an array in the same spare-group that contains all of the standard devices plus a hot spare device. If it finds any, it will move the hot spare to the array that needs it. In our case, if **/dev/md0** were to lose a device, it would look at **/dev/md1** and find the two devices of the array plus a hot spare, and it will move the hot spare device to **/dev/md0** and begin the rebuild process.

Run **mdadm** in daemon mode and have it monitor and scan the arrays:

```
mdadm -F -s -m root@localhost -f
```

The default poll time is 60 seconds, but can be changed using the -d option (e.g., -d 300 would poll every 5 minutes).

Now test out this feature by failing and removing a device from **/dev/md0**:

```
mdadm /dev/md0 -f /dev/sda1 -r /dev/sda1
```

The next time that mdadm polls the devices, it should determine that **/dev/md1** has a spare device, and it should move **/dev/sde1** to **/dev/md0** and rebuild the array. You can then add in **/dev/sda1** and it will become your hot spare device:

```
mdadm /dev/md0 -a /dev/sda1
```

4. USB when the Drivers Aren't Available

By Dominic Duval, Red Hat Certfied Engineer

As a way to save a few valuable pennies on newer PCs, manufacturers are increasingly getting rid of the good old PS/2 keyboard and mouse interfaces. As a result, some recent systems only ship with USB ports to which we need to connect a USB keyboard and mouse.

USB is all well and good, but what if the driver for your USB controller is not loaded? In practice, this is not a problem, as Red Hat loads the ehci- hcd and uhci-hcd drivers automatically at boot time.

There are situations, namely in emergency mode, where the USB drivers won't be available. So you won't even be able to enter a command. This is due to the fact that in emergency mode all drivers need to be provided in the initrd file under /boot, and USB is not there by default. The trick is to add those drivers, so that they will be available earlier. The '**mkinitrd**' command can do precisely that with the '**--with**' argument (this only works under RHEL4):

```
mkinitrd --with=ehci-hcd --with=uhci-hcd /boot/newinitrd-`uname -
r`.img

`uname -r`
```

Add a new entry in your grub.conf file (always do backups!) that points to this new initrd image, and you're done! Your USB keyboard now works in emergency mode.

5. Using Proc

By Steve Bonneville, Red Hat Certfied Engineer

In **/proc**, there are subdirectories for each process running on the system, named based on the PID number of the process. In each of these directories, there is a **fd/** subdirectory that contains files that represent the file descriptors the process currently has open. These files are actually symlinks that point to the actual device, socket, or other file the process currently has open and mapped to that file descriptor.

If you have a program that can read input from a file but not from standard input, or that can write to a file but not to standard output, you may be able to cheat by taking advantage of these special files:

```
/proc/self/fd/0 is standard input of the current process

/proc/self/fd/1 is standard output of the current process

/proc/self/fd/2 is standard error of the current process
```

For example if '**myfilter**' can only read from a file, which it takes as its first argument, you can make it read from standard input instead with:

```
'myfilter /proc/self/fd/0'
```

Another example: **`'cat filename > /proc/self/fd/2'`** sends the contents of filename out standard error instead of standard output.

Whether these tricks will behave in a sane manner will depend on how the process actually handles the file it opens.

6. Growing the Devices in a RAID Array

By Forrest Taylor, Red Hat Certfied Engineer

As hard disk space is ever increasing, you may get replacement drives that are significantly larger than the original devices that they replace, so this tip will show how to increase the size of a RAID array using larger partitions to replace smaller partitions in the original RAID array.

We will assume that you have a RAID 5 array using three partitions (**`/dev/sdb1`**, **`/dev/sdc1`**, and **`/dev/sdd1`**) on **`/dev/md0`**. These partitions are 1 GB each, giving you about 2 GB of usable space. You add new disks and create three partitions (**`/dev/sde1`**, **`/dev/sdf1`**, and **`/dev/sdg1`**) of 5 GB in size. By the end, you should have about 10 GB of usable space.

After you have created the partitions and set the partitions type to **`0xfd`**, you can add these devices to the array. They will become hot spares:

```
mdadm /dev/md0 -a /dev/sde1 /dev/sdf1 /dev/sdg1
```

Fail the original devices one at a time, ensuring that the array rebuilds after each failed device.

Note: *Do not* fail more than one of the original devices without verifying that the array has finished rebuilding. If you fail two devices in a RAID 5 array, you may destroy data!

First, fail and remove the first device, and verify that the array has finished rebuilding:

```
mdadm /dev/md0 -f /dev/sdb1 -r /dev/sdb1

watch cat /proc/mdstat
```

Once it has finished rebuilding, fail the second device:

```
mdadm /dev/md0 -f /dev/sdc1 -r /dev/sdc1

watch cat /proc/mdstat
```

Once it has finished rebuilding, fail the third device:

```
mdadm /dev/md0 -f /dev/sdd1 -r /dev/sdd1

watch cat /proc/mdstat
```

After it has finished rebuilding, you have replaced all of the 1 GB original devices with the new 5 GB devices. However, we are not finished yet. We have two problems: the RAID array is still only using 1 GB of my 5 GB devices, and the filesystem is still 2 GB.

First, grow the RAID array. **mdadm** can grow the RAID array to a certain size, using the **-G** and **-z** options. The **-z** option can take a currently undocumented argument of max, which will resize the array to the maximum available space:

```
mdadm -G /dev/md0 -z max
```

`cat /proc/mdstat` and `mdadm -D /dev/md0` should show that the array is now using a 5 GB device size.

Second, we need to enlarge the filesystem to match. Assuming that you have an ext3 filesystem on /dev/md0, and that you have mounted it, you can increase the size of the filesystem by using **ext2online**:

```
ext2online /dev/md0
```

After that command completes, you should see about 10 GB of usable space.

7. Installing Third-Party RPMs

By Doug Bunger, Red Hat Certified Engineer

After rebuilding a system, it may be necessary to add several additional RPMs. These could be third-party applications or vendor-specific patches. Trying to do an **RPM -i** or **-U** with an ***.rpm** would fail if the process encountered an error. Since the list of RPMs might include packages that were not included with the Red Hat distribution, a **-F** might not work. In such a case, the following could help:

```
find /start/dir -name "*.rpm" \

-exec rpm -Uvh --aid {} \;
```

The first line of the command would get a list of the RPMs available in the directory (**/start/dir**, in the example). The second line would install each RPM in turn. Depending on the nature of the RPMs, it may be necessary to issue the command twice, though the **--aid** option should attempt to resolve dependencies.

8. Partprobe

By Richard Keech, Red Hat Certified Engineer

Many system administrators may be in the habit of re-booting their systems to make partition changes visible to the kernel. With Red Hat Enterprise Linux, this is not usually necessary. The **partprobe** command, from the parted package, informs the kernel about changes to partitions. After all, anything that can help you avoid a re-boot has to be a good thing!

For example:

```
# cat /proc/partitions

major       minor       #blocks     name
3           0           58605120    hda
3           1           200781      hda1
3           2           2040255     hda2
3           3           56364052    hda3
            8           01018880    sda
            8           110224      sda1

      # partprobe

      # cat /proc/partitions

major       minor       #blocks     name
  3           0         58605120     hda
  3           1           200781    hda1
  3           2          2040255    hda2
  3           3         56364052    hda3
  8           0          1018880     sda
  8           1            10224    sda1
  8           2          1008640    sda2
```

9. Pyshell

By Brad Smith, Red Hat Certified Engineer

Python developers: You probably know that the python interpreter can be run in interactive mode, allowing you to quickly try out an approach or prototype a script. Fedora includes an even more powerful version of this tool from an unlikely source. The **`wxPython-common-gtk2-unicode`** package provides files related to the **`wxWindows widget set`** and, more-or-less unrelated to the rest of the package's contents, a tool called pyshell.

Pyshell performs the same basic function as the interactive-mode python interpreter, but with a lot of great bells and whistles. Try importing a module, such as "**`os`**" and then referencing an element of the module:

```
>>> import os

>>> os.
```

When "**`.`**" is typed, up pops a list of every property and method within the "**`os`**" module. You can use the mouse or arrow keys (plus tab-completion) to select what you want. If you select a method, begining the argument list with "**`(`**" pops up a list of the method's accepted arguments and its **`pydoc`** string, where applicable. The best part is that, since **`pyshell`** reads the **`pydoc`** information for each module as it is loaded, this works for any module, including those you've written yourself.

Moving around within **pyshell** can take some getting used to. The up arrow moves you up line by line instead of moving through the interpreter's history like it does in the basic interpreter. **Ctrl+Up** moves through the history. However, the history is in blocks, not lines. So, for example, if you'd defined a class earlier on and then pressed **Ctrl+Up**, when you reached the class in your history, its whole definition would come up. You could then use the arrow keys to move around the definition, making changes. **Ctrl+Enter** even allows you to insert new lines into the definition. When you're done, press **Enter** and the class is re-defined according to your revised code.

Pyshell makes it even easier than before to write and test small applications "on the fly." Once you've got the hang of it, try out the even fancier alternative, pycrust, which integrates a number of tools for browsing structures within the interpreter's memory, viewing output, etc into **pyshell**. Want more? Try **pyalamode**, which has all the features of **pycrust**, plus an integrated version of the **pyalacarte text editor**, for all your cut-and-pasting needs (cutting and pasting into any other editor works fine too).

10. Un-killable Processes

By Johnathan Kupferer, Red Hat Certified Engineer

Before Red Hat Enterprise Linux 4, there really wasn't a good way to handle processes that had entered an uninterruptible sleep waiting on an unresponsive NFS server. This was particularly frustrating because the umount man page promises that "-f" will "Force unmount." This allows an NFS-mounted filesystem to be unmounted if the NFS server is "unreachable." That was how it was supposed to work, with the caveat that the filesystem must have originally been mounted with "soft" or "intr" options. Well, no more. Though the man page doesn't say so, **umount -f** now comes to the rescue and will unmount hard and uninterruptible mounts.

Learn More

Learn more about how you can improve productivity, enhance efficiency, and sharpen your competitive edge.

Check out our complete Red Hat Linux curriculum at www.globalknowledge.com/redhat.

For more information or to register, visit www.globalknowledge.com or call 1-800-COURSES to speak with a sales representative.

Through expert instruction, you will understand key concepts and how to apply them to your specific work situation. Choose from more than 700 courses, delivered through Classrooms, e-Learning, and On-site sessions, to meet your IT and management training needs.