Linux System Administration

Paul Cobbaut

Linux System Administration

Paul Cobbaut

lt-0.970.177

Published Wed 10 Dec 2008 10:06:03 PM CET

Abstract

This book is meant to be used in an instructor-led training. For self-study, the idea is to read this book next to a working Linux computer so you can immediately do every subject, even every command.

This book is aimed towards novice Linux system administrators (and might be interesting and useful for home users that want to know a bit more about their Linux system). However, this book is not meant as an introduction to Linux desktop applications like text editors, browsers, mail clients, multimedia or office applications.

More information and free .pdf available at **http://www.linux-training.be** .

Feel free to contact the authors:

- Paul Cobbaut: paul.cobbaut@gmail.com, http://www.linkedin.com/in/cobbaut

Contributors to the Linux Training project are:

- Serge van Ginderachter: serge@ginsys.be, docbook xml and pdf build scripts; svn hosting
- Hendrik De Vloed: hendrik.devloed@ugent.be, buildheader.pl script

We'd also like to thank our reviewers:

- Wouter Verhelst: wouter@grep.be, http://grep.be
- Geert Goossens: goossens.geert@gmail.com, http://www.linkedin.com/in/geertgoossens
- Elie De Brauwer: elie@de-brauwer.be, http://www.de-brauwer.be
- Christophe Vandeplas: christophe@vandeplas.com, http://christophe.vandeplas.com

Copyright 2007-2008 Paul Cobbaut

Table of Contents

List of Tables

Chapter 1. Disk Management

1.1. Hard disk devices

1.1.1. Terminology

Data is commonly stored on magnetic or optical **disk platters**. The platters are rotated (at high speeds). Data is read by **heads**, which are very close to the surface of the platter, without touching it! The heads are mounted on an arm (sometimes called a comb).

Data is written in concentric circles or **tracks**. Track zero is (usually ?) on the inside. The time it takes to position the head over a certain track is called the **seek time**. Often the platters are stacked on top of each other, hence the set of tracks accessible at a certain position of the comb forms a **cylinder**. Tracks are divided into 512 byte **sectors**, with more unused space (**gap**) between the sectors on the outside of the platter.

When you break down the advertised **access time** of a hard drive, you will notice that most of that time is taken by movement of the heads (about 65%) and **rotational latency** (about 30%).

Random access hard disk devices have an abstraction layer called **block device** to enable formatting in fixed-size (usually 512 bytes) blocks. Blocks can be accessed independent of access to other blocks. You can recognize a block device by the letter b as first character of ls -l.

```
[root@RHEL4b ~]# ls -l /dev/sda*
brw-rw----  1 root disk 8, 0 Aug  4 22:55 /dev/sda
brw-rw----  1 root disk 8, 1 Aug  4 22:55 /dev/sda1
brw-rw----  1 root disk 8, 2 Aug  4 22:55 /dev/sda2
[root@RHEL4b ~]#
```

1.1.2. IDE or SCSI

Actually, the title should be **ATA** or **SCSI**, since IDE is an ATA-compatible device. Most desktops use ATA devices. ATA allows two devices per bus, one **master** and one **slave**. Unless your controller and devices support **cable select**, you have to set this manually with jumpers. With the introduction of **SATA** (Serial ATA), the original ATA was renamed to **Parallel ATA**. Optical drives often use **atapi**, which is an ATA interface using the SCSI communication protocol.

When using the **Small Computer System Interface**, each device gets a unique **SCSI ID**. The SCSI controller also needs a SCSI ID, do not use this ID for a SCSI-attached device. Older 8-bit SCSI is now called **narrow**, whereas 16-bit is **wide**. When the

bus speeds was doubled to 10Mhz, this was known as **fast SCSI**. Doubling to 20Mhz made it **ultra SCSI**. Take a look at http://en.wikipedia.org/wiki/SCSI for more SCSI-standards.

1.1.3. Device Naming

All ATA drives on your system will start with **/dev/hd** followed by a unit letter. The master hdd on the first ATA controller is /dev/hda, the slave is /dev/hdb. For the second controller, the names of the devices are /dev/hdc and /dev/hdd.

Table 1.1. IDE device naming

Controller	Connection	Device Name
IDE0	master	/dev/hda
	slave	/dev/hdb
IDE1	master	/dev/hdc
	slave	/dev/hdd

SCSI drives follow a similar scheme, but all start with **/dev/sd**. When you run out of letters (after /dev/sdz), you can continue with /dev/sdaa and /dev/sdab and so on. (We will see later on that LVM volumes are commonly seen as /dev/md0, /dev/md1 etc)

Below a **sample** of how SCSI devices on a linux can be named. Adding a SCSI disk or RAID controller with a lower SCSI address will change the naming scheme (shifting the higher SCSI addresses one letter further in the alphabet).

Table 1.2. SCSI device naming

Device	SCSI ID	Device Name
Disk 0	0	/dev/sda
Disk 1	1	/dev/sdb
RAID Controller 0	5	/dev/sdc
RAID Controller 1	6	/dev/sdd

To get more information about the naming of devices and their major and minor number, visit http://www.lanana.org/docs/device-list/devices.txt .

1.1.4. Erasing a hard disk

Before selling your old hard disk on the internet, it might be a good idea to really erase it. By simply repartitioning or even after a new mkfs command, some people will still be able to read most of the data on the disk. Although technically the **badblocks** tool is meant to look for bad blocks, you can use it to erase a disk. Since this is really writing to every sector of the disk, it can take a long time!

```
root@RHELv4u2:~# badblocks -ws /dev/sdb
```

```
Testing with pattern 0xaa: done
Reading and comparing: done
Testing with pattern 0x55: done
Reading and comparing: done
Testing with pattern 0xff: done
Reading and comparing: done
Testing with pattern 0x00: done
Reading and comparing: done
```

1.1.5. fdisk

You can start by using **fdisk** to find out what kind of disks are seen by the kernel. Below the result on Debian, with two **ATA-IDE disks** present.

```
root@barry:~# fdisk -l | grep Disk
Disk /dev/hda: 60.0 GB, 60022480896 bytes
Disk /dev/hdb: 81.9 GB, 81964302336 bytes
```

And here an example of **SATA disks** on a laptop with Ubuntu. SATA hard disks are presented to you with the SCSI /dev/sdx notation.

```
root@laika:~# fdisk -l | grep Disk
Disk /dev/sda: 100.0 GB, 100030242816 bytes
Disk /dev/sdb: 100.0 GB, 100030242816 bytes
```

And last but not least, an overview of disks on a RHEL4u3 server with two real 72GB **SCSI disks**. This server is attached to a NAS with four **NAS disks** of half a terabyte. On the NAS disks, four LVM software RAID devices are configured.

```
[root@tsvtl1 ~]# fdisk -l | grep Disk
Disk /dev/sda: 73.4 GB, 73407488000 bytes
Disk /dev/sdb: 73.4 GB, 73407488000 bytes
Disk /dev/sdc: 499.0 GB, 499036192768 bytes
Disk /dev/sdd: 499.0 GB, 499036192768 bytes
Disk /dev/sde: 499.0 GB, 499036192768 bytes
Disk /dev/sdf: 499.0 GB, 499036192768 bytes
Disk /dev/md0: 271 MB, 271319040 bytes
Disk /dev/md2: 21.4 GB, 21476081664 bytes
Disk /dev/md3: 21.4 GB, 21467889664 bytes
Disk /dev/md1: 21.4 GB, 21476081664 bytes
```

You can also use fdisk to obtain information about one specific hard disk device.

```
[root@rhel4 ~]# fdisk -l /dev/sda

Disk /dev/sda: 12.8 GB, 12884901888 bytes
255 heads, 63 sectors/track, 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          13      104391   83  Linux
/dev/sda2              14        1566    12474472+  8e  Linux LVM
```

Later we will use fdisk to do dangerous stuff like creating and deleting partitions.

1.1.6. hdparm

To obtain (or set) information and parameters about an ATA (or SATA) hard disk device, you can use **hdparm**. The -i and -I options will give you even more information about the physical properties of the device.

```
root@laika:~# hdparm /dev/sdb

/dev/sdb:
 IO_support   =  0 (default 16-bit)
 readonly     =  0 (off)
 readahead    = 256 (on)
 geometry     = 12161/255/63, sectors = 195371568, start = 0
```

1.1.7. dmesg

Kernel boot messages can be seen after boot with **dmesg**. Since hard disk devices are detected by the kernel during boot, you can also use dmesg to find information.

```
root@barry:~# dmesg | grep "[hs]d[a-z]"
Kernel command line: root=/dev/hda1 ro
    ide0: BM-DMA at 0xfc00-0xfc07, BIOS settings: hda:DMA, hdb:DMA
    ide1: BM-DMA at 0xfc08-0xfc0f, BIOS settings: hdc:DMA, hdd:DMA
hda: ST360021A, ATA DISK drive
hdb: Maxtor 6Y080L0, ATA DISK drive
hdc: SONY DVD RW DRU-510A, ATAPI CD/DVD-ROM drive
hdd: SONY DVD RW DRU-810A, ATAPI CD/DVD-ROM drive
hda: max request size: 128KiB
hda: 117231408 sectors (60022 MB) w/2048KiB Cache, CHS=65535/16/63, UDMA
 hda: hda1 hda2
hdb: max request size: 128KiB
hdb: 160086528 sectors (81964 MB) w/2048KiB Cache, CHS=65535/16/63, UDMA
 hdb: hdb1 hdb2
hdc: ATAPI 32X DVD-ROM DVD-R CD-R/RW drive, 8192kB Cache, UDMA(33)
hdd: ATAPI 40X DVD-ROM DVD-R CD-R/RW drive, 2048kB Cache, UDMA(33)
...
```

1.1.8. /proc/scsi/scsi

You can also look at the contents of **/proc/scsi/scsi**.

```
root@shaka:~# cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: Adaptec  Model: RAID5            Rev: V1.0
  Type:   Direct-Access                    ANSI SCSI revision: 02
Host: scsi1 Channel: 00 Id: 00 Lun: 00
```

```
Vendor: SEAGATE  Model: ST336605FSUN36G  Rev: 0438
Type:   Direct-Access                    ANSI SCSI revision: 03
root@shaka:~#
```

1.1.9. scsi_info

You can also use **scsi_info**.

```
root@shaka:~# scsi_info /dev/sdb
SCSI_ID="0,0,0"
HOST="1"
MODEL="SEAGATE ST336605FSUN36G"
FW_REV="0438"
root@shaka:~#
```

1.1.10. lsscsi

And even **lsscsi** if it is installed.

```
root@shaka:~# lsscsi
[0:0:0:0]    disk    Adaptec  RAID5            V1.0  /dev/sda
[1:0:0:0]    disk    SEAGATE  ST336605FSUN36G  0438  /dev/sdb
root@shaka:~#
```

1.1.11. Practice hard disk devices

1. Use dmesg to make a list of hard disk devices (ide,ata,sata,scsi) detected at bootup.

2. Use fdisk to find the total size of all hard disk devices on your system.

3. Stop a virtual machine, add a virtual 10 gigabyte SCSI hard disk and a virtual 100 megabyte SCSI hard disk.

4. Use dmesg and fdisk (with grep) to display some information about the new disks.

5. Use badblocks to completely erase the 100 mb hard disk.

6. Look at /proc/scsi/scsi.

1.2. Partitions

1.2.1. About Partitions

Linux requires you to create one or more **partitions** aka **slices**. *Please don't break your head on the difference between a partition and a slice. Different tools have*

different interpretations of which is which. Although partitions reside on the same hard disk device, you can (almost) see them as independent of each other.

A partition's **geometry** and size is usually defined by a starting and ending cylinder (sometimes by head or even sector). Partitions can be of type **primary** (maximum four), **extended** (maximum one) or **logical** (contained within the extended partition). Each partition has a **type field** that contains a code. This determines the computers operating system or the partitions file system.

1.2.2. Partition naming

We saw before that hard disk devices are named /dev/hdx or /dev/sdx with x depending on the hardware configuration. Next is the partition number, starting the count at 1. Hence the four (possible) primary partitions are numbered 1 to 4. Logical partition counting always starts at 5. Thus /dev/hda2 is the second partition on the first ATA hard disk device, and /dev/hdb5 is the first logical partition on the second ATA hard disk device. Same for SCSI, /dev/sdb3 is the third partition on the second SCSI disk.

Table 1.3. Partition naming

Partition Type	naming
Primary (max 4)	1-4
Extended (max 1)	1-4
Logical	5-

1.2.3. fdisk -l

In the **fdisk -l** example below you can see that two partitions exist on /dev/sdb2. The first partition spans 31 cylinders and contains a Linux swap partition. The second partition is much bigger.

```
root@laika:~# fdisk -l /dev/sdb

Disk /dev/sdb: 100.0 GB, 100030242816 bytes
255 heads, 63 sectors/track, 12161 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start      End     Blocks   Id  System
/dev/sdb1             1       31     248976   82  Linux swap / Solaris
/dev/sdb2            32    12161   97434225   83  Linux
root@laika:~#
```

1.2.4. other tools

You might be interested in more GUI-oriented alternatives to fdisk and parted like cfdisk, sfdisk and gparted.

1.2.5. Partitioning new disks

In the example below, we bought a new disk for our system. After the new hardware is properly attached, you can use **fdisk** and **parted** to create the necessary partition(s). This example uses fdisk, but there is nothing wrong with using parted.

First, we check with **fdisk -l** whether Linux can see the new disk. Yes it does, the new disk is seen as /dev/sdb, but it does not have any partitions yet.

```
root@RHELv4u2:~# fdisk -l

Disk /dev/sda: 12.8 GB, 12884901888 bytes
255 heads, 63 sectors/track, 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start      End      Blocks   Id  System
/dev/sda1   *       1        13      104391   83  Linux
/dev/sda2          14      1566    12474472+  8e  Linux LVM

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Disk /dev/sdb doesn't contain a valid partition table
```

Then we create a partition with fdisk on /dev/sdb. First we start the fdisk tool with /dev/sdb as argument. Be very very careful not to partition the wrong disk!!

```
root@RHELv4u2:~# fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI...
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected...
```

Inside the fdisk tool, we can issue the **p** command to see the current disks partition table.

```
Command (m for help): p

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start         End      Blocks   Id  System
```

No partitions exist yet, so we issue **n** to create a new partition. We choose p for primary, 1 for the partition number, 1 for the start cylinder and 14 for the end cylinder.

```
Command (m for help): n
Command action
e   extended
```

```
p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-130, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-130, default 130): 14
```

We can now issue p again to verify our changes, but they are not yet written to disk. This means we can still cancel this operation! But it looks good, so we use **w** to write the changes to disk, and then quit the fdisk tool.

```
Command (m for help): p

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1          14      112423+  83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
root@RHELv4u2:~#
```

Let's verify again with **fdisk -l** to make sure reality fits our dreams. Indeed, the screenshot below now shows a partition on /dev/sdb.

```
root@RHELv4u2:~# fdisk -l

Disk /dev/sda: 12.8 GB, 12884901888 bytes
255 heads, 63 sectors/track, 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          13      104391   83  Linux
/dev/sda2              14        1566    12474472+  8e  Linux LVM

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1          14      112423+  83  Linux
root@RHELv4u2:~#
```

1.2.6. Master Boot Record

The partition table information (primary and extended partitions) is written in the **Master Boot Record** or **MBR**. You can use **dd** to copy the mbr to a file.

This example copies the master boot record from the first SCSI hard disk.

```
dd if=/dev/sda of=/SCSIdisk.mbr bs=512 count=1
```

The same tool can also be used to wipe out all information about partitions on a disk. This example writes zeroes over the master boot record.

```
dd if=/dev/zero of=/dev/sda bs=512 count=1
```

1.2.7. partprobe

Don't forget that after restoring a Master Boot Record with dd, that you need to force the kernel to reread the partition table with **partprobe**. After running partprobe, the partitions can be used again.

```
[root@RHEL5 ~]# partprobe
[root@RHEL5 ~]#
```

1.2.8. logical drives

The partition table does not contain information about logical drives. So the dd backup of the mbr only works for primary and extended partitions. To backup the partition table inlcuding the logical drives, you can use **sfdisk**. The following command will copy the mbs and the logical drive information from /dev/sda to /dev/sdb.

```
sfdisk -d /dev/sda | sfdisk /dev/sdb
```

1.2.9. Practice Partitions

1. Use fdisk and df to display existing partitions and sizes. Compare the output of the two commands.

2. Create a 50MB primary partition on a small disk.

3. Create a 200MB primary partition and two 100MB logical drives on a big disk.

4. Use df and fdisk -l to verify your work.

5. Take a backup of the partition table containing your 200MB primary and 100MB logical drives. Destroy the partitions with fdisk. Then restore your backup.

1.3. File Systems

1.3.1. About file systems

After you are finished partitioning the hard disk, you can put a **file system** on each partition. A file system is a way of organizing files on your partition. Besides

file-based storage, file systems usually include **directories** and **access control**, and contain meta information about files like access times, modification times and file ownership.

The properties (length, character set, ...) of filenames are determined by the file system you choose. Directories are usually implemented as files, you will have to learn how this is implemented! Access control in file systems is tracked by user ownership (and group owner- and membership) in combination with one or more access control lists.

1.3.2. Common file systems

1.3.2.1. ext2 and ext3

Once the most common Linux file systems is the **ext2** (the second extended) file system. A disadvantage is that file system checks on ext2 can take a long time. You will see that ext2 is being replaced by **ext3** on most Linux machines. They are essentially the same, except for the **journaling** which is only present in ext3.

Journaling means that changes are first written to a journal on the disk. The journal is flushed regularly, writing the changes in the file system. Journaling keeps the file system in a consistent state, so you don't need a file system check after an unclean shutdown or power failure.

You can create these file systems with the **/sbin/mkfs** or **/sbin/mke2fs** commands. Use **mke2fs -j** to create an ext3 file system. You can convert an ext2 to ext3 with **tune2fs -j**. You can mount an ext3 file system as ext2, but then you lose the journaling. Do not forget to run **mkinitrd** if you are booting from this device.

1.3.2.2. vfat

The **vfat** file system exists in a couple of forms : FAT12 for floppy disks, **FAT16** on DOS, and **FAT32** for larger disks. The Linux VFAT implementation supports all of these, but vfat lacks a lot of features like security and links. FAT disks can be read by every operating system, and are used a lot for digital cameras, USB sticks and to exchange data between different OS'ses on a home user's computer.

1.3.2.3. ISO 9660

ISO 9660 is the standard format for CD-ROM's. Chances are you will encounter this file system also on your harddisk in the form of images of CD-ROM's (often with the .ISO extension). The ISO 9660 standard limits filenames to the 8.3 format. The Unix world didn't like this, and thus added the **Rock Ridge** extensions, which allows for filenames up to 255 characters and Unix-style file-modes, ownership and symbolic links. Another extensions to ISO 9660 is **Joliet**, which adds 64 unicode

characters to the filename. The **El Torito** standard extends ISO 9660 to be able to boot from CD-ROM's.

1.3.2.4. UDF

Most optical media today (including CD's and DVD's) use **UDF**, the Universal Disk Format.

1.3.2.5. swap

All things considered, swap is not a file system. But to use a partition as a **swap partition** it must be formatted as swap space.

1.3.2.6. others...

You might encounter **reiserfs** on Linux systems, but it is not common on Red Hat. Maybe you will see a **zfs**, or one of the dozen other file systems available.

1.3.3. Putting a file system on a partition

We now have a fresh partition. The system binaries to make file systems can be found with ls.

```
[root@RHEL4b ~]# ls -lS /sbin/mk*
-rwxr-xr-x  3 root root 34832 Apr 24  2006 /sbin/mke2fs
-rwxr-xr-x  3 root root 34832 Apr 24  2006 /sbin/mkfs.ext2
-rwxr-xr-x  3 root root 34832 Apr 24  2006 /sbin/mkfs.ext3
-rwxr-xr-x  3 root root 28484 Oct 13  2004 /sbin/mkdosfs
-rwxr-xr-x  3 root root 28484 Oct 13  2004 /sbin/mkfs.msdos
-rwxr-xr-x  3 root root 28484 Oct 13  2004 /sbin/mkfs.vfat
-rwxr-xr-x  1 root root 20313 Apr 10  2006 /sbin/mkinitrd
-rwxr-x---  1 root root 15444 Oct  5  2004 /sbin/mkzonedb
-rwxr-xr-x  1 root root 15300 May 24  2006 /sbin/mkfs.cramfs
-rwxr-xr-x  1 root root 13036 May 24  2006 /sbin/mkswap
-rwxr-xr-x  1 root root  6912 May 24  2006 /sbin/mkfs
-rwxr-xr-x  1 root root  5905 Aug  3  2004 /sbin/mkbootdisk
[root@RHEL4b ~]#
```

It is time for you to read the manual pages of **mkfs** and **mke2fs**. In the example below, you see the creation of an **ext2 file system** on /dev/sdb1. In real life, you might want to use options like -m0 and -j.

```
root@RHELv4u2:~# mke2fs /dev/sdb1
mke2fs 1.35 (28-Feb-2004)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
28112 inodes, 112420 blocks
5621 blocks (5.00%) reserved for the super user
```

```
First data block=1
Maximum filesystem blocks=67371008
14 block groups
8192 blocks per group, 8192 fragments per group
2008 inodes per group
Superblock backups stored on blocks:
8193, 24577, 40961, 57345, 73729

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 37 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

1.3.4. Tuning a file system

You can use **tune2fs** to list and set file system settings. The first screenshot lists the reserved space for root (which is set at five percent).

```
[root@rhel4 ~]# tune2fs -l /dev/sda1 | grep -i "block count"
Block count:              104388
Reserved block count:     5219
[root@rhel4 ~]#
```

This example changes this value to ten percent. You can use tune2fs while the file system is active, even if it is the root file system (as in this example).

```
[root@rhel4 ~]# tune2fs -m10 /dev/sda1
tune2fs 1.35 (28-Feb-2004)
Setting reserved blocks percentage to 10 (10430 blocks)
[root@rhel4 ~]# tune2fs -l /dev/sda1 | grep -i "block count"
Block count:              104388
Reserved block count:     10430
[root@rhel4 ~]#
```

1.3.5. Checking a file system

The **fsck** command is a front end tool used to check a file system for errors.

```
[root@RHEL4b ~]# ls /sbin/*fsck*
/sbin/dosfsck  /sbin/fsck         /sbin/fsck.ext2  /sbin/fsck.msdos
/sbin/e2fsck   /sbin/fsck.cramfs  /sbin/fsck.ext3  /sbin/fsck.vfat
[root@RHEL4b ~]#
```

The last column in **/etc/fstab** is used to determine whether a file system should be checked at bootup.

```
[paul@RHEL4b ~]$ grep ext /etc/fstab
/dev/VolGroup00/LogVol00   /             ext3    defaults        1 1
LABEL=/boot                /boot         ext3    defaults        1 2
```

```
[paul@RHEL4b ~]$
```

Manually checking a mounted file system results in a warning from fsck.

```
[root@RHEL4b ~]# fsck /boot
fsck 1.35 (28-Feb-2004)
e2fsck 1.35 (28-Feb-2004)
/dev/sda1 is mounted.

WARNING!!!  Running e2fsck on a mounted filesystem may cause
SEVERE filesystem damage.

Do you really want to continue (y/n)? no

check aborted.
[root@RHEL4b ~]#
```

But after unmounting fsck and **e2fsck** can be used to check an ext2 file system.

```
[root@RHEL4b ~]# fsck  /boot
fsck 1.35 (28-Feb-2004)
e2fsck 1.35 (28-Feb-2004)
/boot: clean, 44/26104 files, 17598/104388 blocks
[root@RHEL4b ~]# fsck -p /boot
fsck 1.35 (28-Feb-2004)
/boot: clean, 44/26104 files, 17598/104388 blocks
[root@RHEL4b ~]# e2fsck -p /dev/sda1
/boot: clean, 44/26104 files, 17598/104388 blocks
[root@RHEL4b ~]#
```

1.3.6. Practice File Systems

1. List the filesystems that are known by your system.

2. Create an ext2 filesystem on the 50MB partition.

3. Create an ext3 filesystem on the 4GB primary and one of the 1GB logical drives.

4. Set the reserved space for root on the logical drive to 0 percent.

5. Verify your work with the usual commands.

6. Put a reiserfs on one of the logical drives.

1.4. Mounting

Once you've put a file system on a partition, you can **mount** it. Mounting a file system makes it available for use, usually as a directory. We say **mounting a file system** instead of mounting a partition because we will see later that we can also mount file systems that do not exists on partitions.

1.4.1. Mounting local disks

On all Unix systems, every file and every directory is part of one big file tree. To access a file, you need to know the full path starting from the root directory. When adding a file system to your computer, you need to make it available somewhere in the file tree. The directory where you make a file system available is called a **mount point**. Once mounted, the new file system is accessible to users. The screenshot below shows the creation of a mount point, and the mounting of an ext2 partition on a newly added SCSI disk.

```
root@RHELv4u2:~# mkdir /home/project55
root@RHELv4u2:~# mount -t ext2 /dev/sdb1 /home/project55/
root@RHELv4u2:~# ls /home/project55/
lost+found
root@RHELv4u2:~#
```

Actually the explicit -t ext2 option to set the file system is not always necesarry. The mount command is able to automatically detect a lot of file systems on partitions.

1.4.2. Displaying mounted file systems

To view all mounted file systems, look at the files **/proc/mounts** or **/etc/mtab**. The kernel provides the info in /proc/mount in file form, but /proc/mount does not exist as a file on any hard disk. Looking at /proc/mount is the best way to be sure, since the information comes directly from the kernel. The /etc/mtab file on the other hand is updated by the mount command. Do not edit /etc/mtab manually!

Another way to view all mounts is by issuing the **mount** command without any arguments. The screenshot below pipes the output of these three through grep, to only show our added SCSI disk.

```
root@RHELv4u2:~# cat /proc/mounts | grep /dev/sdb
/dev/sdb1 /home/project55 ext2 rw 0 0
root@RHELv4u2:~# cat /etc/mtab | grep /dev/sdb
/dev/sdb1 /home/project55 ext2 rw 0 0
root@RHELv4u2:~# mount | grep /dev/sdb
/dev/sdb1 on /home/project55 type ext2 (rw)
```

A more user friendly way to look at mounted hard disks is **df**. The df(diskfree) command has the added benefit of showing you the free space on each mounted disk. Like a lot of Linux commands, df supports the **-h** switch to make the output more **human readable**.

```
root@RHELv4u2:~# df
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
11707972   6366996   4746240  58% /
/dev/sda1              101086    9300    86567  10% /boot
none                   127988       0   127988   0% /dev/shm
```

```
/dev/sdb1             108865    1550   101694   2% /home/project55
root@RHELv4u2:~# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
12G  6.1G  4.6G  58% /
/dev/sda1              99M  9.1M   85M  10% /boot
none                  125M     0  125M   0% /dev/shm
/dev/sdb1             107M  1.6M  100M   2% /home/project55
```

1.4.3. Permanent mounts

Until now, we performed all mounts manually. This works nice, until the next reboot. Luckily there is a way to tell your computer to automatically mount certain file systems during boot. This is done using the file system table located in the **/etc/fstab** file. Below is a sample /etc/fstab file.

```
root@RHELv4u2:~# cat /etc/fstab
/dev/VolGroup00/LogVol00 /                ext3    defaults        1 1
LABEL=/boot              /boot            ext3    defaults        1 2
none                     /dev/pts         devpts  gid=5,mode=620  0 0
none                     /dev/shm         tmpfs   defaults        0 0
none                     /proc            proc    defaults        0 0
none                     /sys             sysfs   defaults        0 0
/dev/VolGroup00/LogVol01 swap             swap    defaults        0 0
```

By adding the following two lines, we can automate the mounting of these file systems. The first line is for our freshly added SCSI disk, the second line mounts an NFS share.

```
/dev/sdb1                /home/project55      ext2    defaults    0 0
server12:/mnt/data/iso   /home/iso            nfs     defaults    0 0
```

1.4.4. Disk Usage (du)

The **du** command can summarize disk usage for files and directories. Preventing du to go into subdirectories with the -s option will give you a total for that directory. This option is often used together with -h, so **du -sh** on a mount point gives the total amount used in that partition.

```
root@pasha:~# du -sh /home/reet
881G     /home/reet
```

1.4.5. Disk Free (df)

In the **df -h** example below you can see the size, free space, used gigabytes and percentage and mount point of a partition.

```
root@laika:~# df -h | egrep -e "(sdb2|File)"
Filesystem            Size Used Avail Use% Mounted on
/dev/sdb2              92G   83G  8.6G  91% /media/sdb2
root@laika:~#
```

1.4.6. Practice Mounting File Systems

1. Mount the small 50MB partition on /home/project22.

2. Mount the big primary partition on /mnt, the copy some files to it (everything in /etc). Then mount the partition as read only on /srv/nfs/salesnumbers.

3. Verify your work with fdisk, df, mount. Also look in /etc and /proc to interesting files.

4. Make both mounts permanent, test that it works.

5. What happens when you mount a partition on a directory that contains some files ?

6. What happens when you mount two partitions on the same mountpoint ?

7. Describe the difference between these file searching commands: find, locate, updatedb, whereis, apropos and which.

8. Perform a file system check on the partition mounted at /srv/nfs/salesnumbers.

1.5. UUID and filesystems

1.5.1. About Unique Objects

A **UUID** or **Universally Unique Identifier**is used to uniquely identify objects. This 128bit standard allows anyone to create a unique UUID. Below we use the **vol_id** utility to display the UUID of an ext3 partition on a hard disk.

```
root@laika:~# vol_id --uuid /dev/sda1
825d4b79-ec40-4390-8a71-9261df8d4c82
```

Red Hat Enterprise Linux 5 does not put the **vol_id** command in the PATH. But you can use **tune2fs** or type the path to the vol_id command to display the UUID of a volume.

```
[root@RHEL5 ~]# tune2fs -l /dev/sda1 | grep UUID
Filesystem UUID:          11cfc8bc-07c0-4c3f-9f64-78422ef1dd5c
[root@RHEL5 ~]# /lib/udev/vol_id -u /dev/sda1
```

```
11cfc8bc-07c0-4c3f-9f64-78422ef1dd5c
```

1.5.2. Using UUID in /etc/fstab

You can use the UUID to make sure that a volume is universally uniquely identified in **/etc/fstab**. The device name can change depending on the disk devices that are present at boot time, but a UUID never changes.

First we use **tune2fs** to find the UUID.

```
[root@RHEL5 ~]# tune2fs -l /dev/sdc1 | grep UUID
Filesystem UUID:          7626d73a-2bb6-4937-90ca-e451025d64e8
```

Then we check that it is properly added to **/etc/fstab**, the UUID replaces the variable devicename /dev/sdc1.

```
[root@RHEL5 ~]# grep UUID /etc/fstab
UUID=7626d73a-2bb6-4937-90ca-e451025d64e8 /home/pro42 ext3 defaults 0 0
```

Now we can mount the volume using the mountpoint defined in /etc/fstab.

```
[root@RHEL5 ~]# mount /home/pro42
[root@RHEL5 ~]# df -h | grep 42
/dev/sdc1             397M   11M  366M   3% /home/pro42
```

The real test now, is to remove /dev/sdb from the system, reboot the machine and see what happens. After the reboot, the disk previously known as /dev/sdc is now /dev/sdb.

```
[root@RHEL5 ~]# tune2fs -l /dev/sdb1 | grep UUID
Filesystem UUID:          7626d73a-2bb6-4937-90ca-e451025d64e8
```

And thanks to the UUID in /etc/fstab, the mountpoint is mounted on the same disk as before.

```
[root@RHEL5 ~]# df -h | grep sdb
/dev/sdb1             397M   11M  366M   3% /home/pro42
```

1.5.3. Practice UUID

1. Find the UUID of one of your ext3 partitions with tune2fs and vol_id.

2. USe this UUID in /etc/fstab and test that it works when a disk is removed (and the device name is changed). (You can edit settings in vmware to remove a hard disk.)

1.6. RAID

1.6.1. Hardware or software

Redundant Array of Independent Disks or **RAID** can be set up using hardware or software. Hardware RAID is more expensive, but offers better performance. Software RAID is cheaper and easier to manage, but it uses your CPU and your memory.

1.6.2. RAID levels

1.6.2.1. RAID 0

RAID 0 uses two or more disks, and is often called **striping** (or stripe set, or striped volume). Data is divided in **chunks**, those chunks are evenly spread across every disk in the array. The main advantage of RAID 0 is that you can create **larger drives**. RAID 0 is the only RAID without redundancy.

1.6.2.2. JBOD

JBOD uses two or more disks, and is often called **concatenating** (spanning, spanned set, or spanned volume). Data is written to the first disk, until it is full. Then data is written to the second disk... The main advantage of JBOD (Just a Bunch of Disks) is that you can create **larger drives**. JBOD offers no redundancy.

1.6.2.3. RAID 1

RAID 1 uses exactly two disks, and is often called **mirroring** (or mirror set, or mirrored volume). All data written to the array is written on each disk. The main advantage of RAID 1 is **redundancy**. The main disadvantage is that you lose at least half of your available disk space (in other words, you at least double the cost).

1.6.2.4. RAID 2, 3 and 4 ?

RAID 2 uses bit level striping, RAID 3 byte level, and RAID 4 is the same as RAID 5, but with a dedicated parity disk. This is actually slower than RAID 5, because every write would have to write parity to this one (bottleneck) disk. It is unlikely that you will ever see these RAID levels in production.

1.6.2.5. RAID 5

RAID 5 uses **three** or more disks, each divided into chunks. Every time chunks are written to the array, one of the disks will receive a **parity** chunk. Unlike RAID 4,

the parity chunk will alternate between all disks. The main advantage of this is that RAID 5 will allow for full data recovery in case of **one** hard disk failure.

1.6.2.6. RAID 6

RAID 6 is very similar to RAID 5, but uses two parity chunks. RAID 6 protects against two hard disk failures.

1.6.2.7. RAID 0+1

RAID 0+1 is a mirror(1) of stripes(0). This means you first create two RAID 0 stripe sets, and then you set them up as a mirror set. For example, when you have six 100GB disks, then the stripe sets are each 300GB. Combined in a mirror, this makes 300GB total. RAID 0+1 will survive one disk failure. It will only survive the second disk failure if this disk is in the same stripe set as the previous failed disk.

1.6.2.8. RAID 1+0

RAID 1+0 is a stripe(0) of mirrors(1). For example, when you have six 100GB disks, then you first create three mirrors of 100GB each. You then stripe them together into a 300GB drive. In this example, as long as not all disks in the same mirror fail, it can survive up to three hard disk failures.

1.6.2.9. RAID 50

RAID 5+0 is a stripe(0) of RAID 5 arrays. Suppose you have nine disks of 100GB, then you can create three RAID 5 arrays of 200GB each. You can then combine them into one large stripe set.

1.6.2.10. many others

There are many other nested RAID combinations, like RAID 30, 51, 60, 100, 150, ...

1.6.3. Building a software RAID array

You can do this during the installation with Disk Druid (easy), or afterwards using the commandline (not so easy).

First, you have to attach some disks to your computer. In this scenario, three brand new disks of one gigabyte each are added. Check with **fdisk -l** that they are connected.

```
root@RHELv4u2:~# fdisk -l

Disk /dev/sda: 12.8 GB, 12884901888 bytes
```

```
255 heads, 63 sectors/track, 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          13      104391   83  Linux
/dev/sda2              14        1566    12474472+  8e  Linux LVM

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Disk /dev/sdb doesn't contain a valid partition table

Disk /dev/sdc: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Disk /dev/sdc doesn't contain a valid partition table

Disk /dev/sdd: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Disk /dev/sdd doesn't contain a valid partition table
```

So far so good! Next step is to create a partition of type **fd** on every disk. The fd type is to set the partition as **Linux RAID auto**. Like this screenshot shows.

```
root@RHELv4u2:~# fdisk /dev/sdc
Device contains neither a valid DOS partition table, nor Sun, SGI or \
OSF disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected b\
y w(rite)

Command (m for help): n
Command action
e   extended
p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-130, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-130, default 130):
Using default value 130

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): fd
Changed system type of partition 1 to fd (Linux raid autodetect)

Command (m for help): p

Disk /dev/sdc: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start     End      Blocks   Id  System
/dev/sdc1          1       130     1044193+  fd  Linux raid autodetect
```

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
root@RHELv4u2:~#
```

Now all three disks are ready for RAID, so we have to tell the system what to do with these disks.

```
root@RHELv4u2:~# fdisk -l

Disk /dev/sda: 12.8 GB, 12884901888 bytes
255 heads, 63 sectors/track, 1566 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          13      104391   83  Linux
/dev/sda2              14        1566    12474472+  8e  Linux LVM

Disk /dev/sdb: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start      End      Blocks   Id  System
/dev/sdb1          1        130     1044193+  fd  Linux raid autodetect

Disk /dev/sdc: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start      End      Blocks   Id  System
/dev/sdc1          1        130     1044193+  fd  Linux raid autodetect

Disk /dev/sdd: 1073 MB, 1073741824 bytes
255 heads, 63 sectors/track, 130 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device Boot      Start      End      Blocks   Id  System
/dev/sdd1          1        130     1044193+  fd  Linux raid autodetect
```

The next step used to be *create the RAID table in **/etc/raidtab***. Nowadays, you can just issue the command **mdadm** with the correct parameters. The command below is split on two lines to fit this print, but you should type it on one line, without the backslash (\).

```
root@RHELv4u2:~# mdadm --create /dev/md0 --chunk=64 --level=5 --raid-d\
evices=3 /dev/sdb1 /dev/sdc1 /dev/sdd1
mdadm: array /dev/md0 started.
```

Below a partial screenshot how fdisk -l sees the RAID5

```
root@RHELv4u2:~# fdisk -l

<cut>
```

```
Disk /dev/md0: 2138 MB, 2138308608 bytes
2 heads, 4 sectors/track, 522048 cylinders
Units = cylinders of 8 * 512 = 4096 bytes

Disk /dev/md0 doesn't contain a valid partition table
```

We will use this software RAID 5 array in the next topic, LVM.

1.6.4. /proc/mdstat

The status of the raid devices can be seen in **/proc/mdstat**. This example shows a RAID 5 in the process of rebuilding.

```
[root@RHEL5 ~]# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdg1[3] sdf1[1] sde1[0]
      1677056 blocks level 5, 64k chunk, algorithm 2 [3/2] [UU_]
      [=================>...]  recovery = 89.1% (747952/838528) finish\
=0.0min speed=25791K/sec

unused devices: >none<
```

This example shows an active software RAID 5.

```
[root@RHEL5 ~]# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdg1[2] sdf1[1] sde1[0]
      1677056 blocks level 5, 64k chunk, algorithm 2 [3/3] [UUU]

unused devices: >none<
```

1.6.5. Removing a software RAID

The software raid is visible in /proc/mdstat when active. To remove the raid completely so you can use the disks for other purposes, you first have to stop (de-activate) it with mdadm.

```
mdadm --stop /dev/mdadm
```

When stopped, you can remove the raid with mdadm.

```
mdadm --remove /dev/mdadm
```

The disks can now be repartitioned.

1.6.6. Practice RAID

1. Add three virtual disks of 200MB each to the virtual Red Hat machine.

2. Create a software RAID 5 on the three disks. (It is not necessary to put a filesystem on it)

3. Verify with fdisk and in /proc/ that the RAID exists.

4. (optional) Stop and remove the RAID, unless you want to use it in the next chapter LVM.

Chapter 2. Logical Volume Management

2.1. Introduction to lvm

2.1.1. Problems with standard partitions

There are some problems when working with hard disks and standard partitions. Consider a system with a small and a large hard disk device, partitioned like this. The first disk (/dev/sda) is partitioned in two, the second disk (/dev/sdb) has three partitions.

Table 2.1. Disk Partitioning Example

/dev/sda		**/dev/sdb**			
/dev/sda1	/dev/sda2	/dev/sdb1	/dev/sdb2	/dev/sdb3	unused
/boot	/	/var	/home	/project42	
ext2	ext3	ext2	reiserfs	ext3	

In the example above, consider the options when you want to enlarge the space available for project42. What can you do ? The solution will always force you to unmount the filesystem, take a backup of the data, remove and recreate partitions, and then restore the data and remount the file system.

2.1.2. Solution with lvm

Using **lvm** will create a virtual layer between the mounted file systems and the hardware devices. This virtual layer will allow for an administrator to enlarge a mounted file system in use. When lvm is properly used, then there is no need to unmount the file system to enlarge it.

Table 2.2. LVM Example

/dev/sda		**/dev/sdb**			
Volume Group					
/boot	/	/var	/home	/project42	
ext2	ext3	ext2	reiserfs	ext3	

2.1.3. About lvm

Most lvm implementations support **physical storage grouping**, **logical volume resizing** and **data migration**.

Physical storage grouping is a fancy name for grouping multiple physical devices (hard disks) into a logical mass storage device. To enlarge this physical group, hard disks or even single partitions can be added at a later time. The size of LVM volumes on this physical group is independent of the individual size of the components. The total size of the group is the limit.

One of the nicest features of LVM is the logical volume resizing. You can increase the size of an LVM volume, sometimes even without any downtime. Additionally, you can migrate data away from a failing hard disk device.

2.2. LVM Terminology

2.2.1. Physical Volume (pv)

A **physical volume** is a disk, a partition or a (hardware or software) RAID device. All these devices can become a member of a **Volume Group**.

2.2.2. Volume Group (vg)

A **Volume Group** is an abstraction layer between **Physical Devices** and **Logical Volumes**.

2.2.3. Logical Volume (lv)

A **Logical Volume** is created in a **Volume Group**. Logical Volumes that contain a file system can be mounted. The use of logical volumes is similar to the use of partitions (both are standard block devices) and is accomplished with the same standard commands (mkfs, mount, fsck, df, ...).

2.3. Verifying existing Physical Volumes

2.3.1. lvmdiskscan

To get a list of block devices that can be used with LVM, use **lvmdiskscan**. The example below uses grep to limit the result to SCSI devices.

```
[root@RHEL5 ~]# lvmdiskscan | grep sd
  /dev/sda1                [      101.94 MB]
  /dev/sda2                [       15.90 GB] LVM physical volume
  /dev/sdb                 [      409.60 MB]
  /dev/sdc                 [      409.60 MB]
  /dev/sdd                 [      409.60 MB] LVM physical volume
```

```
  /dev/sde1               [       95.98 MB]
  /dev/sde5               [      191.98 MB]
  /dev/sdf                [      819.20 MB] LVM physical volume
  /dev/sdg1               [      818.98 MB]
[root@RHEL5 ~]#
```

2.3.2. pvs

The easiest way to verify whether devices are known to lvm is with the **pvs** command. The screenshot below shows that only /dev/sda2 is currently known for use with LVM. It shows that /dev/sda2 is part of Volgroup00 and is almost 16GB in size. It also shows /dev/sdc and /dev/sdd as part of vg33. The device /dev/sdb is knwon to lvm, but not linked to any Volume Group.

```
[root@RHEL5 ~]# pvs
  PV         VG         Fmt  Attr PSize   PFree
  /dev/sda2  VolGroup00 lvm2 a-    15.88G      0
  /dev/sdb              lvm2 --   409.60M 409.60M
  /dev/sdc   vg33       lvm2 a-   408.00M 408.00M
  /dev/sdd   vg33       lvm2 a-   408.00M 408.00M
[root@RHEL5 ~]#
```

2.3.3. pvscan

The **pvscan** command will scan all disks for existing Physical Volumes. The information is similar to pvs, plus you get a line with total sizes.

```
[root@RHEL5 ~]# pvscan
  PV /dev/sdc    VG vg33         lvm2 [408.00 MB / 408.00 MB free]
  PV /dev/sdd    VG vg33         lvm2 [408.00 MB / 408.00 MB free]
  PV /dev/sda2   VG VolGroup00   lvm2 [15.88 GB / 0    free]
  PV /dev/sdb                    lvm2 [409.60 MB]
  Total: 4 [17.07 GB] / in use: 3 [16.67 GB] / in no VG: 1 [409.60 MB]
[root@RHEL5 ~]#
```

2.3.4. pvdisplay

Use **pvdisplay** to get more information about physical volumes. You can also use **pvdisplay** without an argument to display information about all physical (lvm) volumes.

```
[root@RHEL5 ~]# pvdisplay /dev/sda2
  --- Physical volume ---
  PV Name               /dev/sda2
  VG Name               VolGroup00
  PV Size               15.90 GB / not usable 20.79 MB
  Allocatable           yes (but full)
  PE Size (KByte)       32768
```

```
  Total PE              508
  Free PE               0
  Allocated PE          508
  PV UUID               TobYfp-Ggg0-Rf8r-xtLd-5XgN-RSPc-8vkTHD

[root@RHEL5 ~]#
```

2.4. Verifying existing Volume Groups

2.4.1. vgs

Similar to **pvs** is the use of **vgs** to display a quick overview of all volume groups. There is only one volume group in the screenshot below, it is named VolGroup00 and is almost 16GB in size.

```
[root@RHEL5 ~]# vgs
  VG         #PV #LV #SN Attr   VSize  VFree
  VolGroup00   1   2   0 wz--n- 15.88G    0
[root@RHEL5 ~]#
```

2.4.2. vgscan

The **vgscan** command will scan all disks for existing Volume Groups. It will also update the **/etc/lvm/.cache** file. This file contains a list of all current lvm devices.

```
[root@RHEL5 ~]# vgscan
  Reading all physical volumes.  This may take a while...
  Found volume group "VolGroup00" using metadata type lvm2
[root@RHEL5 ~]#
```

LVM will run the vgscan automatically at bootup, so if you add hotswap devices, then you will need to run vgscan to update /etc/lvm/.cache with the new devices.

2.4.3. vgdisplay

The **vgdisplay** command will give you more detailed information about a volume group (or about all volume groups if you omit the argument).

```
[root@RHEL5 ~]# vgdisplay VolGroup00
  --- Volume group ---
  VG Name               VolGroup00
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  3
  VG Access             read/write
```

```
  VG Status             resizable
  MAX LV                0
  Cur LV                2
  Open LV               2
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               15.88 GB
  PE Size               32.00 MB
  Total PE              508
  Alloc PE / Size       508 / 15.88 GB
  Free  PE / Size       0 / 0
  VG UUID               qsXvJb-71qV-9l7U-ishX-FobM-qptE-VXmKIg

[root@RHEL5 ~]#
```

2.5. Verifying existing Logical Volumes

2.5.1. lvs

Use **lvs** for a quick look at all existing logical volumes. Below you can see two logical volumes named LogVol00 and LogVol01.

```
[root@RHEL5 ~]# lvs
  LV       VG         Attr   LSize  Origin Snap%  Move Log Copy%
  LogVol00 VolGroup00 -wi-ao 14.88G
  LogVol01 VolGroup00 -wi-ao  1.00G
[root@RHEL5 ~]#
```

2.5.2. lvscan

The **lvscan** command will scan all disks for existing Logical Volumes.

```
[root@RHEL5 ~]# lvscan
  ACTIVE            '/dev/VolGroup00/LogVol00' [14.88 GB] inherit
  ACTIVE            '/dev/VolGroup00/LogVol01' [1.00 GB] inherit
[root@RHEL5 ~]#
```

2.5.3. lvdisplay

More detailed information about logical volumes is available through the **lvdisplay(1)** command.

```
[root@RHEL5 ~]# lvdisplay VolGroup00/LogVol01
  --- Logical volume ---
  LV Name                /dev/VolGroup00/LogVol01
  VG Name                VolGroup00
```

```
  LV UUID                RnTGK6-xWsi-t530-ksJx-7cax-co5c-A1KlDp
  LV Write Access        read/write
  LV Status              available
  # open                 1
  LV Size                1.00 GB
  Current LE             32
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           253:1

[root@RHEL5 ~]#
```

2.6. Manage Physical Volumes

2.6.1. pvcreate

Use the **pvcreate** command to add devices to lvm. This example shows how to add a disk (or hardware RAID device) to lvm.

```
[root@RHEL5 ~]# pvcreate /dev/sdb
  Physical volume "/dev/sdb" successfully created
[root@RHEL5 ~]#
```

This example shows how to add a partition to lvm.

```
[root@RHEL5 ~]# pvcreate /dev/sdc1
  Physical volume "/dev/sdc1" successfully created
[root@RHEL5 ~]#
```

You can also add multiple disks or partitions as target to pvcreate. This example adds three disks to lvm.

```
[root@RHEL5 ~]# pvcreate /dev/sde /dev/sdf /dev/sdg
  Physical volume "/dev/sde" successfully created
  Physical volume "/dev/sdf" successfully created
  Physical volume "/dev/sdg" successfully created
[root@RHEL5 ~]#
```

2.6.2. pvremove

Use the **pvremove** command to remove physical volumes from lvm. The devices may not be in use.

```
[root@RHEL5 ~]# pvremove /dev/sde /dev/sdf /dev/sdg
  Labels on physical volume "/dev/sde" successfully wiped
  Labels on physical volume "/dev/sdf" successfully wiped
```

```
  Labels on physical volume "/dev/sdg" successfully wiped
[root@RHEL5 ~]#
```

2.6.3. pvresize

When you used fdisk to resize a partition on a disk, then you must use **pvresize** to make lvm recognize the new size of the physical volume that represents this partition.

```
[root@RHEL5 ~]# pvresize /dev/sde1
  Physical volume "/dev/sde1" changed
  1 physical volume(s) resized / 0 physical volume(s) not resized
```

2.6.4. pvchange

With **pvchange** you can prevent the allocation of a Physical Volume in a new Volume Group or Logical Volume. This can be useful if you plan to remove a Physical Volume.

```
[root@RHEL5 ~]# pvchange -xn /dev/sdd
  Physical volume "/dev/sdd" changed
  1 physical volume changed / 0 physical volumes not changed
[root@RHEL5 ~]#
```

To revert your previous decision, this example shows you how te re-enable the Physical Volume to allow allocation.

```
[root@RHEL5 ~]# pvchange -xy /dev/sdd
  Physical volume "/dev/sdd" changed
  1 physical volume changed / 0 physical volumes not changed
[root@RHEL5 ~]#
```

2.6.5. pvmove

With **pvmove** you can move Logical Volumes from within a Volume Group to another Physical Volume. This must be done before removing a Physical Volume.

```
[root@RHEL5 ~]# pvs | grep vg1
  /dev/sdf   vg1        lvm2 a-   816.00M      0
  /dev/sdg   vg1        lvm2 a-   816.00M 816.00M
[root@RHEL5 ~]# pvmove /dev/sdf
  /dev/sdf: Moved: 70.1%
  /dev/sdf: Moved: 100.0%
[root@RHEL5 ~]# pvs | grep vg1
  /dev/sdf   vg1        lvm2 a-   816.00M 816.00M
  /dev/sdg   vg1        lvm2 a-   816.00M      0
```

2.7. Manage Volume Groups

2.7.1. vgcreate

Use the **vgcreate** command to create a volume group. You can immediately name all the physical volumes that span the volume group.

```
[root@RHEL5 ~]# vgcreate vg42 /dev/sde /dev/sdf
  Volume group "vg42" successfully created
[root@RHEL5 ~]#
```

2.7.2. vgextend

Use the **vgextend** command to extend an existing volume group with a physical volume.

```
[root@RHEL5 ~]# vgextend vg42 /dev/sdg
  Volume group "vg42" successfully extended
[root@RHEL5 ~]#
```

2.7.3. vgremove

Use the **vgremove** command to remove volume groups from lvm. The volume groups may not be in use.

```
[root@RHEL5 ~]# vgremove vg42
  Volume group "vg42" successfully removed
[root@RHEL5 ~]#
```

2.7.4. vgreduce

Use the **vgreduce** command to remove a Physical Volume from the Volume Group.

The following example adds Physical Volume /dev/sdg to the vg1 Volume Group using vgextend. And then removes it again using vgreduce.

```
[root@RHEL5 ~]# pvs | grep sdg
  /dev/sdg              lvm2 --   819.20M 819.20M
[root@RHEL5 ~]# vgextend vg1 /dev/sdg
  Volume group "vg1" successfully extended
[root@RHEL5 ~]# pvs | grep sdg
  /dev/sdg   vg1        lvm2 a-   816.00M 816.00M
[root@RHEL5 ~]# vgreduce vg1 /dev/sdg
  Removed "/dev/sdg" from volume group "vg1"
[root@RHEL5 ~]# pvs | grep sdg
  /dev/sdg              lvm2 --   819.20M 819.20M
```

2.7.5. vgchange

Use the **vgchange** command to change parameters of a Volume Group.

This example shows how to prevent Physical Volumes from being added or removed to the Volume Group vg1.

```
[root@RHEL5 ~]# vgchange -xn vg1
  Volume group "vg1" successfully changed
[root@RHEL5 ~]# vgextend vg1 /dev/sdg
  Volume group vg1 is not resizeable.
```

You can also use vgchange to change most other properties of a Volume Group. This example changes the maximum nhumber of Logical Volumes and maximum number of Physical Volumes that vg1 can serve.

```
[root@RHEL5 ~]# vgdisplay vg1 | grep -i max
  MAX LV                0
  Max PV                0
[root@RHEL5 ~]# vgchange -l16 vg1
  Volume group "vg1" successfully changed
[root@RHEL5 ~]# vgchange -p8 vg1
  Volume group "vg1" successfully changed
[root@RHEL5 ~]# vgdisplay vg1 | grep -i max
  MAX LV                16
  Max PV                8
```

2.7.6. vgmerge

Merging two Volume Groups into one is done with **vgmerge**. The following example merges vg2 into vg1, keeping all the properties of vg1.

```
[root@RHEL5 ~]# vgmerge vg1 vg2
  Volume group "vg2" successfully merged into "vg1"
[root@RHEL5 ~]#
```

2.8. Manage Logical Volumes

2.8.1. lvcreate

Use the **lvcreate** command to create Logical Volumes in a Volume Group. This example creates an 8GB Logical Volume in Volume Group vg42.

```
[root@RHEL5 ~]# lvcreate -L5G vg42
```

```
  Logical volume "lvol0" created
[root@RHEL5 ~]#
```

As you can see, lvm automatically names the Logical Volume **lvol0**. The next example creates a 200MB Logical Volume named MyLV in Volume Group vg42.

```
[root@RHEL5 ~]# lvcreate -L200M -nMyLV vg42
  Logical volume "MyLV" created
[root@RHEL5 ~]#
```

The next example does the same thing, but with different syntax.

```
[root@RHEL5 ~]# lvcreate --size 200M -n MyLV vg42
  Logical volume "MyLV" created
[root@RHEL5 ~]#
```

This example creates a Logical Volume that occupies 10 percent of the Volume Group.

```
[root@RHEL5 ~]# lvcreate -l 10%VG -n MyLV2 vg42
  Logical volume "MyLV2" created
[root@RHEL5 ~]#
```

This example creates a Logical Volume that occupies 30 percent of the remaining free space in the Volume Group.

```
[root@RHEL5 ~]# lvcreate -l 30%FREE -n MyLV3 vg42
  Logical volume "MyLV3" created
[root@RHEL5 ~]#
```

2.8.2. lvremove

Use the **lvremove** command to remove Logical Volumes from a Volume Group. Removing a Logical Volume requires the name of the Volume Group.

```
[root@RHEL5 ~]# lvremove vg42/MyLV
Do you really want to remove active logical volume "MyLV"? [y/n]: y
  Logical volume "MyLV" successfully removed
[root@RHEL5 ~]#
```

Removing multiple Logical Volumes will request confirmation for each individual volume.

```
[root@RHEL5 ~]# lvremove vg42/MyLV vg42/MyLV2 vg42/MyLV3
Do you really want to remove active logical volume "MyLV"? [y/n]: y
  Logical volume "MyLV" successfully removed
Do you really want to remove active logical volume "MyLV2"? [y/n]: y
  Logical volume "MyLV2" successfully removed
```

```
Do you really want to remove active logical volume "MyLV3"? [y/n]: y
  Logical volume "MyLV3" successfully removed
[root@RHEL5 ~]#
```

2.8.3. lvextend

Extending the volume is easy with **lvextend**. This example extends a 200MB Logical Volume with 100 MB.

```
[root@RHEL5 ~]# lvdisplay /dev/vg2/lvol0 | grep Size
  LV Size                200.00 MB
[root@RHEL5 ~]# lvextend -L +100 /dev/vg2/lvol0
  Extending logical volume lvol0 to 300.00 MB
  Logical volume lvol0 successfully resized
[root@RHEL5 ~]# lvdisplay /dev/vg2/lvol0 | grep Size
  LV Size                300.00 MB
```

The next example creates a 100MB Logical Volume, and then extends it to 500MB.

```
[root@RHEL5 ~]# lvcreate --size 100M -n extLV vg42
  Logical volume "extLV" created
[root@RHEL5 ~]# lvextend -L 500M vg42/extLV
  Extending logical volume extLV to 500.00 MB
  Logical volume extLV successfully resized
[root@RHEL5 ~]#
```

This example doubles the size of a Logical Volume.

```
[root@RHEL5 ~]# lvextend -l+100%LV vg42/extLV
  Extending logical volume extLV to 1000.00 MB
  Logical volume extLV successfully resized
[root@RHEL5 ~]#
```

2.8.4. lvrename

Renaming a Logical Volume is done with **lvrename**. This example renames extLV to bigLV in the vg42 Volume Group.

```
[root@RHEL5 ~]# lvrename vg42/extLV vg42/bigLV
  Renamed "extLV" to "bigLV" in volume group "vg42"
[root@RHEL5 ~]#
```

2.9. Example: Using lvm

This example shows how you can use a device (in this case /dev/sdc, but it could have been /dev/sdb or any other disk or partition) with lvm, how to create a volume group (vg) and how to create and use a logical volume (vg/lvol0).

First thing to do, is create physical volumes that can join the volume group with **pvcreate**. This command makes a disk or partition available for use in Volume Groups. The screenshot shows how to present the SCSI Disk device to LVM.

```
root@RHEL4:~# pvcreate /dev/sdc
Physical volume "/dev/sdc" successfully created
```

Note for home users: lvm will work fine when using the complete disk, but another operating system on the same computer will not recognize lvm and will mark the disk as being empty! You can avoid this by creating a partition that spans the whole disk, then run pvcreate on the partition instead of the disk.

Then **vgcreate** creates a volume group using one device. Note that more devices could be added to the volume group.

```
root@RHEL4:~# vgcreate vg /dev/sdc
Volume group "vg" successfully created
```

The last step **lvcreate** creates a logical volume.

```
root@RHEL4:~# lvcreate --size 500m vg
Logical volume "lvol0" created
```

The logical volume /dev/vg/lvol0 can now be formatted with ext2, and mounted for normal use.

```
root@RHELv4u2:~# mke2fs -m0 -j /dev/vg/lvol0
mke2fs 1.35 (28-Feb-2004)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
128016 inodes, 512000 blocks
0 blocks (0.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67633152
63 block groups
8192 blocks per group, 8192 fragments per group
2032 inodes per group
Superblock backups stored on blocks:
8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 37 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
root@RHELv4u2:~# mkdir /home/project10
root@RHELv4u2:~# mount /dev/vg/lvol0 /home/project10/
root@RHELv4u2:~# df -h | grep proj
/dev/mapper/vg-lvol0  485M   11M  474M   3% /home/project10
```

A logical volume is very similar to a partition, it can be formatted with a file system, and can be mounted so users can access it.

2.10. Example: Extend a Logical Volume

A logical volume can be extended without unmounting the file system. Whether or not a volume can be extended depends on the file system it uses. Volumes that are mounted as vfat or ext2 cannot be extended, so in the example here we use the ext3 file system.

The fdisk command shows us newly added scsi-disks that will serve our lvm volume. This volume will then be extended. First, take a look at these disks.

```
[root@RHEL5 ~]# fdisk -l | grep sd[bc]
Disk /dev/sdb doesn't contain a valid partition table
Disk /dev/sdc doesn't contain a valid partition table
Disk /dev/sdb: 1181 MB, 1181115904 bytes
Disk /dev/sdc: 429 MB, 429496320 bytes
```

You already know how to partition a disk, below the first disk is partitioned (in one big primary partition), the second disk is left untouched.

```
[root@RHEL5 ~]# fdisk -l | grep sd[bc]
Disk /dev/sdc doesn't contain a valid partition table
Disk /dev/sdb: 1181 MB, 1181115904 bytes
/dev/sdb1               1         143     1148616   83  Linux
Disk /dev/sdc: 429 MB, 429496320 bytes
```

You also know how to prepare disks for lvm with **pvcreate**, and how to create a volume group with **vgcreate**. This example adds both the partitiond disk and the untouched disk to the volume group named **vg2**.

```
[root@RHEL5 ~]# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
[root@RHEL5 ~]# pvcreate /dev/sdc
  Physical volume "/dev/sdc" successfully created
[root@RHEL5 ~]# vgcreate vg2 /dev/sdb1 /dev/sdc
  Volume group "vg2" successfully created
```

You can use **pvdisplay** to verify that both the disk and the partition belong to the volume group.

```
[root@RHEL5 ~]# pvdisplay | grep -B1 vg2
  PV Name               /dev/sdb1
  VG Name               vg2
--
  PV Name               /dev/sdc
  VG Name               vg2
```

And you are familiar both with the **lvcreate** command to create a small logical volume and the **mke2fs** command to put ext2 on it.

```
[root@RHEL5 ~]# lvcreate --size 200m vg2
  Logical volume "lvol0" created
[root@RHEL5 ~]# mke2fs -m20 -j /dev/vg2/lvol0
...
```

As you see, we end up with a mounted logical volume that according to **df** is almost 200 megabyte in size.

```
[root@RHEL5 ~]# mkdir /home/resizetest
[root@RHEL5 ~]# mount /dev/vg2/lvol0 /home/resizetest/
[root@RHEL5 ~]# df -h | grep resizetest
                      194M  5.6M  149M   4% /home/resizetest
```

Extending the volume is easy with **lvextend**.

```
[root@RHEL5 ~]# lvextend -L +100 /dev/vg2/lvol0
  Extending logical volume lvol0 to 300.00 MB
  Logical volume lvol0 successfully resized
```

But as you can see, there is a small problem: it appears that df is not able to display the extended volume in its full size. This is because the filesystem is only set for the size of the volume before the extension was added.

```
[root@RHEL5 ~]# df -h | grep resizetest
                      194M  5.6M  149M   4% /home/resizetest
```

With **lvdisplay** however we can see that the volume is indeed extended.

```
[root@RHEL5 ~]# lvdisplay /dev/vg2/lvol0 | grep Size
  LV Size                300.00 MB
```

To finish the extension, you need **resize2fs** to span the filesystem over the full size of the logical volume.

```
[root@RHEL5 ~]# resize2fs /dev/vg2/lvol0
resize2fs 1.39 (29-May-2006)
Filesystem at /dev/vg2/lvol0 is mounted on /home/resizetest; on-line re\
sizing required
Performing an on-line resize of /dev/vg2/lvol0 to 307200 (1k) blocks.
The filesystem on /dev/vg2/lvol0 is now 307200 blocks long.
```

Congratulations, you just successfully expanded a logical volume.

```
[root@RHEL5 ~]# df -h | grep resizetest
                      291M  6.1M  225M   3% /home/resizetest
[root@RHEL5 ~]#
```

2.11. Example: Resize a Physical Volume

This is a humble demonstration of how to resize a physical Volume with lvm (after you resize it with fdisk). The demonstration starts with a 100MB partition named /dev/sde1. We used fdisk to create it, and to verify the size.

```
[root@RHEL5 ~]# fdisk -l 2>/dev/null | grep sde1
/dev/sde1               1         100      102384   83  Linux
[root@RHEL5 ~]#
```

Now we can use pvcreate to create the Physical Volume, followed by pvs to verify the creation.

```
[root@RHEL5 ~]# pvcreate /dev/sde1
  Physical volume "/dev/sde1" successfully created
[root@RHEL5 ~]# pvs | grep sde1
  /dev/sde1             lvm2 --    99.98M  99.98M
[root@RHEL5 ~]#
```

The next step is ti use fdisk to enlarge the partition (actually deleting it and then recreating /dev/sde1 with more cylinders).

```
[root@RHEL5 ~]# fdisk /dev/sde

Command (m for help): p

Disk /dev/sde: 858 MB, 858993152 bytes
64 heads, 32 sectors/track, 819 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sde1               1         100      102384   83  Linux

Command (m for help): d
Selected partition 1

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4):
Value out of range.
Partition number (1-4): 1
First cylinder (1-819, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-819, default 819): 200

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
[root@RHEL5 ~]#
```

When we now use fdisk and pvs to verify the size of the partition and the Physical Volume, then there is a size difference. LVM is still using the old size.

```
[root@RHEL5 ~]# fdisk -l 2>/dev/null | grep sde1
/dev/sde1               1         200      204784   83  Linux
[root@RHEL5 ~]# pvs | grep sde1
  /dev/sde1             lvm2 --    99.98M  99.98M
```

```
[root@RHEL5 ~]#
```

Executing pvresize on the Physical Volume will make lvm aware of the size change of the partition. The correct size can be displayed with pvs.

```
[root@RHEL5 ~]# pvresize /dev/sde1
  Physical volume "/dev/sde1" changed
  1 physical volume(s) resized / 0 physical volume(s) not resized
[root@RHEL5 ~]# pvs | grep sde1
  /dev/sde1            lvm2 --   199.98M 199.98M
[root@RHEL5 ~]#
```

2.12. Example: Mirror a Logical Volume

We start by creating three physical volumes for lvm. Then we verify the creation and the size with pvs. Three physical disks because lvm uses two disks for the mirror and a third disk for the mirror log!

```
[root@RHEL5 ~]# pvcreate /dev/sdb /dev/sdc /dev/sdd
  Physical volume "/dev/sdb" successfully created
  Physical volume "/dev/sdc" successfully created
  Physical volume "/dev/sdd" successfully created
[root@RHEL5 ~]# pvs
  PV         VG         Fmt  Attr PSize   PFree
  /dev/sdb              lvm2 --   409.60M 409.60M
  /dev/sdc              lvm2 --   409.60M 409.60M
  /dev/sdd              lvm2 --   409.60M 409.60M
```

Then we create the Volume Group and verify again with pvs. Notice how the three physical volumes now belong to vg33, and how the size is rounded down (in steps of the extent size, here 4MB).

```
[root@RHEL5 ~]# vgcreate vg33 /dev/sdb /dev/sdc /dev/sdd
  Volume group "vg33" successfully created
[root@RHEL5 ~]# pvs
  PV         VG         Fmt  Attr PSize   PFree
  /dev/sda2  VolGroup00 lvm2 a-    15.88G      0
  /dev/sdb   vg33       lvm2 a-   408.00M 408.00M
  /dev/sdc   vg33       lvm2 a-   408.00M 408.00M
  /dev/sdd   vg33       lvm2 a-   408.00M 408.00M
[root@RHEL5 ~]#
```

The last step is to create the Logical Volume with **lvcreate**. Notice the **-m 1** switch to create one mirror. Notice also the change in free space in all three Physical Volumes!

```
[root@RHEL5 ~]# lvcreate --size 300m -n lvmir -m 1 vg33
  Logical volume "lvmir" created
[root@RHEL5 ~]# pvs
  PV         VG         Fmt  Attr PSize   PFree
  /dev/sda2  VolGroup00 lvm2 a-    15.88G      0
  /dev/sdb   vg33       lvm2 a-   408.00M 108.00M
```

```
  /dev/sdc   vg33       lvm2 a-   408.00M 108.00M
  /dev/sdd   vg33       lvm2 a-   408.00M 404.00M
```

You can see the copy status of the mirror with lvs. It currently shows a 100 percent copy.

```
[root@RHEL5 ~]# lvs vg33/lvmir
  LV    VG   Attr   LSize   Origin Snap%  Move Log        Copy%
  lvmir vg33 mwi-ao 300.00M                    lvmir_mlog 100.00
```

2.13. Example: Snapshot a Logical Volume

A snapshot is a virtual copy of all the data at a point in time on a volume. A snapshot Logical Volume will retain a copy of all changed files of the snapshotted Logical Volume.

The example below creates a snapshot of the bigLV Logical Volume.

```
[root@RHEL5 ~]# lvcreate -L100M -s -n snapLV vg42/bigLV
  Logical volume "snapLV" created
[root@RHEL5 ~]#
```

You can see with lvs that the snapshot snapLV is indeed a snapshot of bigLV. Moments after taking the snapshot, there are few changes to bigLV (0.02 percent).

```
[root@RHEL5 ~]# lvs
  LV       VG         Attr   LSize   Origin Snap%  Move Log Copy%
  bigLV    vg42       owi-a- 200.00M
  snapLV   vg42       swi-a- 100.00M bigLV    0.02
[root@RHEL5 ~]#
```

But after using bigLV for a while, more changes are done. This means the snapshot volume has to keep more original data (10.22 percent).

```
[root@RHEL5 ~]# lvs | grep vg42
  bigLV    vg42       owi-ao 200.00M
  snapLV   vg42       swi-a- 100.00M bigLV   10.22
[root@RHEL5 ~]#
```

You can now use regular backup tools (dump, tar, cpio, ...) to take a backup of the snapshot Logical Volume. This backup will contain all data as it existed on bigLV at the time the snapshot was taken. When the backup is done, you can remove the snapshot.

```
[root@RHEL5 ~]# lvremove vg42/snapLV
Do you really want to remove active logical volume "snapLV"? [y/n]: y
  Logical volume "snapLV" successfully removed
[root@RHEL5 ~]#
```

2.14. Practice LVM

1. Create a volume group that contains a complete disk and a partition on another disk.

2. Create two logical volumes (a small one and a bigger one) in this volumegroup. Format them wih ext3, mount them and copy some files to them.

3. Verify usage with fdisk, mount, pvs, vgs, lvs, pvdisplay, vgdisplay, lvdisplay and df. Does fdisk give you any information about lvm?

4. Enlarge the small logical volume by 50 percent, and verify your work!

5. Take a look at other commands that start with vg* , pv* or lv*.

6. Create a mirror and a striped Logical Volume.

7. Convert a linear logical volume to a mirror.

8. Convert a mirror logical volume to a linear.

9. Create a snapshot of a Logical Volume, take a backup of the snapshot. Then delete some files on the Logical Volume, then restore your backup.

10. Move your volume group to another disk (keep the Logical Volumes mounted).

11. If time permits, split a Volume Group with vgsplit, then merge it again with vgmerge.

Chapter 3. Booting Linux

3.1. Booting the system

Booting the system starts (once the hardware is powered on and configured) with a bootloader.

There are a variety of boot loaders available, most common on intel architecture is **GRUB**, which is replacing **Lilo** in many places. When installing Linux on SPARC architecture, you can choose **Silo**, Itanium systems can use **ELILO**, IBM S/390 and zSeries use **z/IPL** and PowerPC architectures use **YABOOT** (which means Yet Another boot loader).

Once the Linux kernel is loaded, the bootloader turns control over to it. From that moment on, the kernel is in control of the system. After discussing bootloaders, we continue with the init system that starts all the daemons.

3.2. GRUB

3.2.1. GRand Unified Bootloader

The most common bootloader on linux systems today is **grub**. On almost all intel based systems grub is replacing **lilo** (LInux LOader). Even Solaris switched to grub on x86 architecture.

One of the big advantages of rub over lilo is the capability to change the configuration during boot (by pressing e to edit the boot commandline).

3.2.2. menu.lst

Grub's configuration file is called **menu.lst** (old versions used **grub.conf**) and is located in /boot/grub. The screenshot below shows (part of) a typical grub configuration file.

```
paul@barry:~$ cat /boot/grub/menu.lst
default  0
timeout  5
color cyan/blue white/blue

title  Debian GNU/Linux, kernel 2.6.17-2-686
root  (hd0,0)
kernel  /boot/vmlinuz-2.6.17-2-686 root=/dev/hda1 ro
initrd  /boot/initrd.img-2.6.17-2-686
savedefault
boot

title  Debian GNU/Linux, kernel 2.6.15-1-686 (recovery mode)
root  (hd0,0)
```

```
kernel  /boot/vmlinuz-2.6.15-1-686 root=/dev/hda1 ro single
initrd  /boot/initrd.img-2.6.15-1-686
savedefault
boot

title  Debian GNU/Linux, kernel 2.6.15-1-686
root  (hd0,0)
kernel  /boot/vmlinuz-2.6.15-1-686 root=/dev/hda1 ro
initrd  /boot/initrd.img-2.6.15-1-686
savedefault
...
```

In the screenshot, you see some parameters at the top like **default** and **timeout**, followed by three **stanzas**. Each stanza is shown as a seperate choice on the bootmenu, and can be booted by grub by selecting it and pressing enter. The stanzas are automatically numbered, starting with 0. So when the default switch marks 0, then the first stanza will be booted, unless another choice is manually selected at boot time. Setting the default switch to 1 will boot the second stanza.

The **timeout** switch defines the amount of seconds a user has to make a choice in the bootmenu. In this example, the timeout is set to five seconds, so grub will wait five seconds before booting the first stanza.

Another interesting switch is **fallback**. This parameter allows you to set a backup stanza in case the first one fails.

3.2.3. Stanza commands

The **title** command serves as a description of the stanza that is visible in the bootmenu. It can be anything you like. In this example it describes the distro and kernel version.

```
title  Debian GNU/Linux, kernel 2.6.17-2-686
root  (hd0,0)
kernel  /boot/vmlinuz-2.6.17-2-686 root=/dev/hda1 ro
initrd  /boot/initrd.img-2.6.17-2-686
```

The **root** command will point to the hard disk to use. (hd0) is the first hard disk device, (hd1) is the second hard disk device. (hd0,0) is the first partition on the first disk, (hd0,1) is the second partition on that disk.

The **kernel** command point to the kernel(file) that grub needs to load. And **initrd** points to an initial RAM disk to accompany the kernel at system boot before mounting / .

3.2.4. Chainloading

Chainloading refers to grub loading another operating systems bootloader. The **chainloader** switch receives one opetion: the number of sectors to read and boot. For DOS one sector is enough, so a grub entry for a DOS or OS/2 partition might look like this. Note that MS-DOS requires the boot/root partition to be active!

```
title MS-DOS 6.22
root  (hd0,1)
makeactive
chainloader +1
```

3.2.5. Installing grub

Run the **grub-install** to install grub. The command requires a destination for overwriting the boot sector or mbr.

```
# grub-install /dev/hda
```

3.3. Lilo

3.3.1. Linux Loader

Lilo used to be the most used Linux bootloader, but is steadily being replaced in x86 with grub.

3.3.2. lilo.conf

Here is an example of a typical **lilo.conf** file. The **delay** switch receives a number in tenths of a second. So the delay below is three seconds, not thirty!

```
boot = /dev/hda
delay = 30

image = /boot/vmlinuz
  root = /dev/hda1
  label = Red Hat 5.2

image = /boot/vmlinuz
  root = /dev/hda2
  label = S.U.S.E. 8.0

other = /dev/hda4
  table = /dev/hda
  label = MS-DOS 6.22
```

The configration file shows three example stanzas. The first one boots Red Hat from the first partition on the first disk (hda1). The second stanza boots Suse 8.0 from the next partition. The last one loads MS-DOS.

3.4. Booting

The kernel receives system control from the bootloader. After a while the kernel starts the **init daemon**. The init daemon has **PID 1**. Many unix and linux systems use(d) init scripts to start daemons in the **System V release 4** style (explained in detail below).

But this synchronous (one after the other) method of starting daemons is slow, and although slow booting is not a problem on servers where uptime is measured in years, the recent uptake of linux on the desktop results in user complaints. To improve linux (and Solaris) startup speed, **Canonical** has developed **upstart** (first used in Ubuntu) and **Sun** has developed **Service Management Facility** for Solaris 10. Both systems are asynchronous and can replace the SysV init scripts. There is also an ongoing effort to create **initng** (init next generation).

3.5. Daemons

A **daemon** is a process that runs in background, without a link to a GUI or terminal. Daemons are usually started at system boot, and stay alive until the system shuts down. In more recent technical writings, daemons are often refered to as **services**.

Unix **daemons** are not to be confused with demons. Evi Nemeth, co-author of the UNIX System Administration Handbook has the following to say about daemons:

Many people equate the word "daemon" with the word "demon", implying some kind of satanic connection between UNIX and the underworld. This is an egregious misunderstanding. "Daemon" is actually a much older form of "demon"; daemons have no particular bias towards good or evil, but rather serve to help define a person's character or personality. The ancient Greeks' concept of a "personal daemon" was similar to the modern concept of a "guardian angel" ....

3.6. Init

3.6.1. /etc/inittab

After the kernel, **/sbin/init** is started with PID 1. Init will read its configuration file **/etc/inittab**. In that file, it will look for the value of initdefault (3 in the screenshot below).

```
[paul@rhel4 ~]$ grep ^id /etc/inittab
id:3:initdefault:
```

3.6.2. Runlevel

This number indicates the default **runlevel**. Some linuxes have a brief description of runlevels in /etc/inittab, like here on Red Hat Enterprise Linux 4.

```
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you don't have network)
#   3 - Full multiuser mode
```

```
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
```

Runlevel 0 means the system is shutting down. Runlevel 1 is used for troubleshooting, only the root user can log on, and only at the console. Runlevel 3 is typical for servers, whereas runlevel 5 is typical for desktops (graphical logon). Besides runlevels 0, 1 and 6, the use may vary depending on the distribution. Some Debian and derived linux systems have full network and GUI logon on runlevels 2 to 5. So always verify the proper meaning of runlevels on your system.

3.6.3. sysinit

Independent of the runlevel, init will run the **/etc/rc.d/rc.sysinit** script (**/etc/init.d/rcS** on debian). This script does a lot of things : setting environment, populating /etc/mtab, mounting file systems, starting swap and more.

```
[paul ~]$ egrep -e"^# Ini" -e"^# Sta" -e"^# Che" /etc/rc.d/rc.sysinit
# Check SELinux status
# Initialize hardware
# Start the graphical boot, if necessary; /usr may not be mounted yet...
# Initialiaze ACPI bits
# Check filesystems
# Start the graphical boot, if necessary and not done yet.
# Check to see if SELinux requires a relabel
# Initialize pseudo-random number generator
# Start up swapping.
# Initialize the serial ports.
[paul ~]$
```

That **egrep** command could also have been written with **grep** like this : grep "^# \(Ini \|Sta\|Che\)". The screenshot above was made on Red Hat Enterprise Linux 4.

3.6.4. rc scripts

Init will continue to read /etc/inittab and meets this section on debian linux.

```
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
```

(on Red Hat Enterprise Linux it is identical except init.d is rc.d.

```
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
```

```
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

In both cases, this means that init will start the rc script with as only parameter the runlevel. Actually /etc/inittab has fields seperated by colons. The second field determines the runlevel in which this line should be executed. So in both cases, only one line of the seven will be executed, depending on the runlevel set by initdefault.

When you take a look in the relevant **/etc/rc3.d** directory, which is real on debian and a symbolic link to **/etc/rc.d/rc3.d** on Red Hat, then you will see a lot of (links to) scripts who's name start with either uppercase K or uppercase S. When entering a runlevel, scripts with uppercase S are started in alphabetical order with "start" as the only parameter. When leaving a runlevel, the same happens for scripts starting with K. All this is done by the rc script.

3.6.5. Power and Ctrl-Alt-Del

When rc is finished starting all those scripts, init will continue to read /etc/inittab. It will read commands on what to execute in case of **powerfailure**, powerok and **Ctrl-Alt-Delete**. The init process never stops keeping an eye on power failures and that triple key combo.

The relevant part on Red Hat Enterprise Linux.

```
[paul@RHEL4b ~]$ grep "\(^c\|^p\)" /etc/inittab
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
pf::powerfail:/sbin/shutdown -f -h +2 "PowerFailure;System Shutting Down"
pr:12345:powerokwait:/sbin/shutdown -c "PowerRestored;Shutdown Cancelled"
```

And very similar on Debian Etch.

```
paul@barry:~$ grep "\(^c\|^p\)" /etc/inittab
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
```

3.6.6. getty

Almost at the end of /etc/inittab, there is a section to start and **respawn** several mingetty's.

```
[root@RHEL4b ~]# grep getty /etc/inittab
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
```

```
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
[root@RHEL4b ~]#
```

A mingetty will display a message on a virtual console and allow you to type a userid and sends that info to the login program. The login program will verify whether that user exists in /etc/passwd and prompt for (and verify) a password. If the password is correct, login passes control to the shell listed in /etc/passwd.

So the getty's are started by init, and watched until they die (user exit's the shell and is logged out). When this happens, the init daemon will respawn a new mingetty. So even if you kill the mingetty's, they will be reborn automatically.

```
[root@RHEL4b ~]# ps fax |grep mingetty
 3038 tty1     Ss+    0:00 /sbin/mingetty tty1
 3039 tty2     Ss+    0:00 /sbin/mingetty tty2
 3040 tty3     Ss+    0:00 /sbin/mingetty tty3
 3041 tty4     Ss+    0:00 /sbin/mingetty tty4
 3042 tty5     Ss+    0:00 /sbin/mingetty tty5
 3043 tty6     Ss+    0:00 /sbin/mingetty tty6
[root@RHEL4b ~]# kill 3038 3039 3040 3041 3042 3043
[root@RHEL4b ~]# ps fax |grep mingetty
 4774 tty1     Ss+    0:00 /sbin/mingetty tty1
 4884 tty2     Ss+    0:00 /sbin/mingetty tty2
 4974 tty3     Ss+    0:00 /sbin/mingetty tty3
 5026 tty4     Ss+    0:00 /sbin/mingetty tty4
 5073 tty5     Ss+    0:00 /sbin/mingetty tty5
 5098 tty6     Ss+    0:00 /sbin/mingetty tty6
[root@RHEL4b ~]#
```

You can disable a mingetty for a certain tty by removing the runlevel from the second field in its line in /etc/inittab. Don't forget to tell init about the change of its configuration file with **kill -1 1**.

3.7. Starting and stopping daemons

The K and S scripts usually are links to the real scripts in **/etc/init.d** or **/etc/rc.d/init.d**. These can also be used when the system is running to start and stop daemons (or services). Most of them accept the following parameters: start, stop, restart, status.

```
root@laika:~# /etc/init.d/samba restart
 * Stopping Samba daemons...                                    [ OK ]
 * Starting Samba daemons...                                    [ OK ]
root@laika:~#
```

You can achieve the same result on Red Hat and derived linuxes with the **service** command.

```
[root@RHEL4b ~]# service smb restart
```

```
Shutting down SMB services:                                [  OK  ]
Shutting down NMB services:                                [  OK  ]
Starting SMB services:                                     [  OK  ]
Starting NMB services:                                     [  OK  ]
[root@RHEL4b ~]#
```

3.8. Display the runlevel

You can see your current runlevel with the **runlevel** or **who -r** commands.

The runlevel command is typical linux and will output the previous and the current runlevel. If there was no previous runlevel, then it will mark it with the letter N.

```
[root@RHEL4b ~]# runlevel
N 3
```

The history of who -r dates back to older unixes, and it still works on linux.

```
[root@RHEL4b ~]# who -r
         run-level 3  Jul 28 09:15                   last=S
```

3.9. Changing the runlevel

You can switch to another runlevel with the **telinit** command. On Linux **/sbin/telinit** is usually a hard link to /sbin/init.

3.10. more info

You might also want to take a look at **chkconfig**, **update-rc.d**, **shutdown**, **poweroff** and passing **init=/bin/bash** to the kernel.

3.11. Practice

1. Take a copy of the kernel, initrd and System.map files in /boot in /boot, naming them 3.0 instead of 2.6. Then add a stanza in grub for this 3.0 files.

2. Change /etc/inittab so that only two mingetty's are respawned. Kill the other mingetty's and verify that they don't come back.

3. Use the Red Hat Enterprise Linux virtual machine. Go to runlevel 5, display the current and previous runlevel, then go back to runlevel 3.

4. Is the sysinit script on your computers setting or changing the PATH environment variable ?

5. Write a script that acts like a daemon script in /etc/init.d/. It should have a case statement to act on start/stop/restart and status. Test the script!

6. Have your script started automatically in runlevel 3, test that it works. If it works, also try stopping it in a runlevel.

7. If time permits, use chkconfig to setup your script in runlevels 2 and 3.

3.12. Solutions

1. The files should look like in this screenshot, verify the names in /boot. You could do a reboot and test your stanza.

```
[root@RHEL5 ~]# grep 3.0.18 /boot/grub/menu.lst
title Red Hat Enterprise Linux Server (3.0.18)
 kernel /vmlinuz-3.0.18 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
 initrd /initrd-3.0.18.img
[root@RHEL5 ~]#
```

2. Killing the mingetty's will result in init respawning them. You can edit /etc/inittab so it looks like the screenshot below. Don't forget to also run kill -1 1.

```
[root@RHEL5 ~]# grep tty /etc/inittab
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2:respawn:/sbin/mingetty tty3
4:2:respawn:/sbin/mingetty tty4
5:2:respawn:/sbin/mingetty tty5
6:2:respawn:/sbin/mingetty tty6
[root@RHEL5 ~]#
```

3. use the telinit (or init) and runlevel commands

4. On Red Hat, grep for PATH in /etc/rc.sysinit, on Debian/Ubuntu check /etc/rc.local. The answer is probably no, but on RHEL5 the rc.sysinit script does set the HOSTNAME variable.

```
[root@RHEL5 etc]# grep HOSTNAME rc.sysinit
```

5. The script could look something like this.

```
#!/bin/bash
# /etc/init.d/pold
#

# always runs
touch /var/lock/pold
```

```
case "$1" in
  start)
    echo -n "Starting pold..."
    sleep 1;
    echo "done."
    ;;
  stop)
    echo -n "Stopping pold..."
    sleep 1;
    echo "done."
    ;;
  *)
    echo "Usage: /etc/init.d/pold {start|stop}"
    exit 1
    ;;
esac

exit 0
```

Chapter 4. Linux Kernel

4.1. about the Linux kernel

4.1.1. kernel versions

In 1991 Linux Torvalds wrote (the first version of) the Linux kernel. He put it online, and other people started contributing code. Over 4000 individuals contributed source code to the latest kernel release (version 2.6.27 in November 2008).

Major Linux kernel versions used to come in even and odd numbers. Versions **2.0**, **2.2**, **2.4** and **2.6** are considered stable kernel versions. Whereas **2.1**, **2.3** and **2.5** were unstable (read development) versions. Since the release of 2.6.0 in January 2004, all development has been done in the 2.6 tree. There is currently no v2.7.x and according to Linus the even/stable vs odd/development scheme is abandoned forever.

4.1.2. uname -r

To see your current Linux kernel version, issue the **uname -r** command as shown below.

This first example shows Linux major version **2.6** and minor version **24**. The rest **-22-generic** is specific to the distribution (Ubuntu in this case).

```
paul@laika:~$ uname -r
2.6.24-22-generic
```

The same command on Red Hat Enterprise Linux shows an older kernel (2.6.18) with **-92.1.17.el5** being specific to the distribution.

```
[paul@RHEL52 ~]$ uname -r
2.6.18-92.1.17.el5
```

4.2. Linux kernel source

4.2.1. ftp.kernel.org

The home of the Linux kernel source is **ftp.kernel.org**. It contains all official releases of the Linux kernel source code from 1991. It provides free downloads over http, ftp and rsync of all these releases, as well as changelogs and patches. More information can be otained on the website **www.kernel.org**.

Anyone can anonymously use an ftp client to access ftp.kernel.org

```
paul@laika:~$ ftp ftp.kernel.org
Connected to pub3.kernel.org.
220 Welcome to ftp.kernel.org.
Name (ftp.kernel.org:paul): anonymous
331 Please specify the password.
Password:
230-       Welcome to the
230-
230-   LINUX KERNEL ARCHIVES
230-       ftp.kernel.org
```

All the Linux kernel versions are located in the pub/linux/kernel/ directory.

```
ftp> ls pub/linux/kernel/v*
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxrwsr-x    2 536      536          4096 Mar 20  2003 v1.0
drwxrwsr-x    2 536      536         20480 Mar 20  2003 v1.1
drwxrwsr-x    2 536      536          8192 Mar 20  2003 v1.2
drwxrwsr-x    2 536      536         40960 Mar 20  2003 v1.3
drwxrwsr-x    3 536      536         16384 Feb 08  2004 v2.0
drwxrwsr-x    2 536      536         53248 Mar 20  2003 v2.1
drwxrwsr-x    3 536      536         12288 Mar 24  2004 v2.2
drwxrwsr-x    2 536      536         24576 Mar 20  2003 v2.3
drwxrwsr-x    5 536      536         28672 Dec 02 08:14 v2.4
drwxrwsr-x    4 536      536         32768 Jul 14  2003 v2.5
drwxrwsr-x    7 536      536        110592 Dec 05 22:36 v2.6
226 Directory send OK.
ftp>
```

4.2.2. /usr/src

On your local computer, the kernel source is located in **/usr/src**. Note though that the structure inside /usr/src might be different depending on the distribution that you are using.

First let's take a look at **/usr/src on Debian**. There appear to be two versions of the complete Linux source code there. Looking for a specific file (e1000_main.c) with find reveals it's exact location.

```
paul@barry:~$ ls -l /usr/src/
drwxr-xr-x 20 root root     4096 2006-04-04 22:12 linux-source-2.6.15
drwxr-xr-x 19 root root     4096 2006-07-15 17:32 linux-source-2.6.16
paul@barry:~$ find /usr/src -name e1000_main.c
/usr/src/linux-source-2.6.15/drivers/net/e1000/e1000_main.c
/usr/src/linux-source-2.6.16/drivers/net/e1000/e1000_main.c
```

This is very similar to **/usr/src on Ubuntu**, except there is only one kernel here (and it is newer).

```
paul@laika:~$ ls -l /usr/src/
drwxr-xr-x 23 root root     4096 2008-11-24 23:28 linux-source-2.6.24
paul@laika:~$ find /usr/src -name "e1000_main.c"
```

```
/usr/src/linux-source-2.6.24/drivers/net/e1000/e1000_main.c
```

Now take a look at **/usr/src on Red Hat Enterprise Linux**.

```
[paul@RHEL52 ~]$ ls -l /usr/src/
drwxr-xr-x 5 root root 4096 Dec  5 19:23 kernels
drwxr-xr-x 7 root root 4096 Oct 11 13:22 redhat
```

We will have to dig a little deeper to find the kernel source on Red Hat!

```
[paul@RHEL52 ~]$ cd /usr/src/redhat/BUILD/
[paul@RHEL52 BUILD]$ find . -name "e1000_main.c"
./kernel-2.6.18/linux-2.6.18.i686/drivers/net/e1000/e1000_main.c
```

4.2.3. downloading the kernel source

4.2.3.1. Debian

Installing the kernel source on Debian is really simple with **aptitude install linux-source**. You can do a search for all linux-source packeges first, like in this screenshot.

```
root@barry:~# aptitude search linux-source
v   linux-source           -
v   linux-source-2.6       -
id  linux-source-2.6.15    - Linux kernel source for version 2.6.15
i   linux-source-2.6.16    - Linux kernel source for version 2.6.16
p   linux-source-2.6.18    - Linux kernel source for version 2.6.18
p   linux-source-2.6.24    - Linux kernel source for version 2.6.24
```

And then use **aptitude install** to download and install the Debian Linux kernel source code.

```
root@barry:~# aptitude install linux-source-2.6.24
```

When the aptitude is finished, you will see a new file named **/usr/src/linux-source-<version>.tar.bz2**

```
root@barry:/usr/src# ls -lh
drwxr-xr-x 20 root root 4.0K 2006-04-04 22:12 linux-source-2.6.15
drwxr-xr-x 19 root root 4.0K 2006-07-15 17:32 linux-source-2.6.16
-rw-r--r--  1 root root  45M 2008-12-02 10:56 linux-source-2.6.24.tar.bz2
```

4.2.3.2. Ubuntu

Ubuntu is based on Debian and also uses **aptitude**, so the task is very similar.

```
root@laika:~# aptitude search linux-source
i   linux-source           - Linux kernel source with Ubuntu patches
v   linux-source-2.6       -
i A linux-source-2.6.24    - Linux kernel source for version 2.6.24
root@laika:~# aptitude install linux-source
```

And when aptitude finishes, we end up with a **/usr/src/linux-source-<version>.tar.bz** file.

```
oot@laika:~# ll /usr/src
total 45M
-rw-r--r--  1 root root  45M 2008-11-24 23:30 linux-source-2.6.24.tar.bz2
```

4.2.3.3. Red Hat Enterprise Linux

The Red Hat kernel source is located on the fourth source cdrom. The file is called **kernel-2.6.9-42.EL.src.rpm** (example for RHELv4u4). It is also available online at ftp://ftp.redhat.com/pub/redhat/linux/enterprise/5Server/en/os/SRPMS/ (example for RHEL5).

To download the kernel source on RHEL, use this long wget command (on one line, without the trailing \).

```
wget ftp://ftp.redhat.com/pub/redhat/linux/enterprise/5Server/en/os/\
SRPMS/kernel-`uname -r`.src.rpm
```

When the wget download is finished, you end up with a 60M .rpm file.

```
[root@RHEL52 src]# ll
total 60M
-rw-r--r-- 1 root root  60M Dec  5 20:54 kernel-2.6.18-92.1.17.el5.src.rpm
drwxr-xr-x 5 root root 4.0K Dec  5 19:23 kernels
drwxr-xr-x 7 root root 4.0K Oct 11 13:22 redhat
```

We will need to perform some more steps before this can be used as kernel source code.

First, we issue the **rpm -i kernel-2.6.9-42.EL.src.rpm** command to install this Red Hat package.

```
[root@RHEL52 src]# ll
total 60M
-rw-r--r-- 1 root root  60M Dec  5 20:54 kernel-2.6.18-92.1.17.el5.src.rpm
drwxr-xr-x 5 root root 4.0K Dec  5 19:23 kernels
drwxr-xr-x 7 root root 4.0K Oct 11 13:22 redhat
[root@RHEL52 src]# rpm -i kernel-2.6.18-92.1.17.el5.src.rpm
```

The we move to the SPECS directory and perform an **rpmbuild**.

```
[root@RHEL52 ~]# cd /usr/src/redhat/SPECS
[root@RHEL52 SPECS]# rpmbuild -bp -vv --target=i686 kernel-2.6.spec
```

The rpmbuild command put the RHEL Linux kernel source code in **/usr/src/redhat/BUILD/kernel-<version>/**.

```
[root@RHEL52 kernel-2.6.18]# pwd
/usr/src/redhat/BUILD/kernel-2.6.18
[root@RHEL52 kernel-2.6.18]# ll
total 20K
drwxr-xr-x  2 root root 4.0K Dec  6  2007 config
-rw-r--r--  1 root root 3.1K Dec  5 20:58 Config.mk
drwxr-xr-x 20 root root 4.0K Dec  5 20:58 linux-2.6.18.i686
drwxr-xr-x 19 root root 4.0K Sep 20  2006 vanilla
drwxr-xr-x  8 root root 4.0K Dec  6  2007 xen
```

4.3. kernel boot files

4.3.1. vmlinuz

The **vmlinuz** file in /boot is the compressed kernel.

```
paul@barry:~$ ls -lh /boot | grep vmlinuz
-rw-r--r-- 1 root root 1.2M 2006-03-06 16:22 vmlinuz-2.6.15-1-486
-rw-r--r-- 1 root root 1.1M 2006-03-06 16:30 vmlinuz-2.6.15-1-686
-rw-r--r-- 1 root root 1.3M 2008-02-11 00:00 vmlinuz-2.6.18-6-686
paul@barry:~$
```

4.3.2. initrd

The kernel uses **initrd** (an initial RAM disk) at boot time. The initrd is mounted before the kernel loads, and can contain additional drivers and modules. It is a **compressed cpio archive**, so you can look at the contents in this way.

```
root@RHELv4u4:/boot# mkdir /mnt/initrd
root@RHELv4u4:/boot# cp initrd-2.6.9-42.0.3.EL.img TMPinitrd.gz
root@RHELv4u4:/boot# gunzip TMPinitrd.gz
root@RHELv4u4:/boot# file TMPinitrd
TMPinitrd: ASCII cpio archive (SVR4 with no CRC)
root@RHELv4u4:/boot# cd /mnt/initrd/
root@RHELv4u4:/mnt/initrd# cpio -i | /boot/TMPinitrd
4985 blocks
root@RHELv4u4:/mnt/initrd# ls -l
total 76
drwxr-xr-x  2 root root 4096 Feb  5 08:36 bin
drwxr-xr-x  2 root root 4096 Feb  5 08:36 dev
drwxr-xr-x  4 root root 4096 Feb  5 08:36 etc
-rwxr-xr-x  1 root root 1607 Feb  5 08:36 init
drwxr-xr-x  2 root root 4096 Feb  5 08:36 lib
```

```
drwxr-xr-x  2 root root 4096 Feb  5 08:36 loopfs
drwxr-xr-x  2 root root 4096 Feb  5 08:36 proc
lrwxrwxrwx  1 root root    3 Feb  5 08:36 sbin -> bin
drwxr-xr-x  2 root root 4096 Feb  5 08:36 sys
drwxr-xr-x  2 root root 4096 Feb  5 08:36 sysroot
root@RHELv4u4:/mnt/initrd#
```

4.3.3. System.map

The **System.map** contains the symbol table and changes with every kernel compile. The symbol table is also present in **/proc/kallsyms** (pre 2.6 kernels name this file /proc/ksyms).

```
root@RHELv4u4:/boot# head System.map-`uname -r`
00000400 A __kernel_vsyscall
0000041a A SYSENTER_RETURN_OFFSET
00000420 A __kernel_sigreturn
00000440 A __kernel_rt_sigreturn
c0100000 A _text
c0100000 T startup_32
c01000c6 t checkCPUtype
c0100147 t is486
c010014e t is386
c010019f t L6
root@RHELv4u4:/boot# head /proc/kallsyms
c0100228 t _stext
c0100228 t calibrate_delay_direct
c0100228 t stext
c0100337 t calibrate_delay
c01004db t rest_init
c0100580 t do_pre_smp_initcalls
c0100585 t run_init_process
c01005ac t init
c0100789 t early_param_test
c01007ad t early_setup_test
root@RHELv4u4:/boot#
```

4.3.4. .config

The last file copied to the /boot directory is the kernel configuration used for compilation. This file is not necessary in the /boot directory, but it is common practice to put a copy there. It allows you to recompile a kernel, starting from the same configuration as an existing working one.

4.4. Linux kernel modules

4.4.1. about kernel modules

The Linux kernel is a monolithic kernel with loadable modules. These modules contain parts of the kernel used typically for device drivers, file systems and network

protocols. Most of the time the necesarry kernel modules are loaded automatically and dynamically without administrator interaction.

4.4.2. /lib/modules

The modules are stored in the **/lib/modules/<kernel-version>** directory. There is a seperate directory for each kernel that was compiled for your system.

```
paul@laika:~$ ll /lib/modules/
total 12K
drwxr-xr-x 7 root root 4.0K 2008-11-10 14:32 2.6.24-16-generic
drwxr-xr-x 8 root root 4.0K 2008-12-06 15:39 2.6.24-21-generic
drwxr-xr-x 8 root root 4.0K 2008-12-05 12:58 2.6.24-22-generic
```

4.4.3. <module>.ko

The file containing the modules usually ends in **.ko**. This screenshot shows the location of the isdn module files.

```
paul@laika:~$ find /lib/modules -name isdn.ko
/lib/modules/2.6.24-21-generic/kernel/drivers/isdn/i4l/isdn.ko
/lib/modules/2.6.24-22-generic/kernel/drivers/isdn/i4l/isdn.ko
/lib/modules/2.6.24-16-generic/kernel/drivers/isdn/i4l/isdn.ko
```

4.4.4. lsmod

To see a list of currently loaded modules, use **lsmod**. You see the name of each loaded module, the size, the use count, and the names of other modules using this one.

```
[root@RHEL52 ~]# lsmod | head -5
Module                  Size  Used by
autofs4                24517  2
hidp                   23105  2
rfcomm                 42457  0
l2cap                  29505  10 hidp,rfcomm
```

4.4.5. /proc/modules

Naturally, the same information is present in **/proc/modules**. Actually **lsmod** only reads and reformats the output of /proc/modules.

```
[root@RHEL52 ~]# head -5 /proc/modules
autofs4 24517 2 - Live 0xe09bd000
hidp 23105 2 - Live 0xe09d3000
rfcomm 42457 0 - Live 0xe0ac1000
```

```
l2cap 29505 10 hidp,rfcomm, Live 0xe0ab8000
bluetooth 53797 5 hidp,rfcomm,l2cap, Live 0xe09e4000
```

4.4.6. insmod

Kernel modules can be manually loaded with the **insmod** command. This is a very simple (and obsolete) way of loading modules. The screenshot shows **insmod** loading the fat module (for fat file system support).

```
root@barry:/lib/modules/2.6.17-2-686# pwd
/lib/modules/2.6.17-2-686
root@barry:/lib/modules/2.6.17-2-686# lsmod | grep fat
root@barry:/lib/modules/2.6.17-2-686# insmod kernel/fs/fat/fat.ko
root@barry:/lib/modules/2.6.17-2-686# lsmod | grep fat
fat                    46588  0
```

insmod is not detecting dependencies, so it fails to load the isdn module (because the isdn module depends on the slhc module).

```
[root@RHEL52 drivers]# pwd
/lib/modules/2.6.18-92.1.18.el5/kernel/drivers
[root@RHEL52 kernel]# insmod isdn/i4l/isdn.ko
insmod: error inserting 'isdn/i4l/isdn.ko': -1 Unknown symbol in module
```

4.4.7. modinfo

As you can see in the screenshot of **modinfo** below, the isdn module depends in the slhc module.

```
[root@RHEL52 drivers]# modinfo isdn/i4l/isdn.ko | head -6
filename:       isdn/i4l/isdn.ko
license:        GPL
author:         Fritz Elfert
description:    ISDN4Linux: link layer
srcversion:     99650346E708173496F6739
depends:        slhc
```

4.4.8. modprobe

The big advantage of **modprobe** over **insmod** is that modprobe will load all necessary modules, whereas insmod requires manual loading of depedencies. Another advantage is that you don't need to point to the filename with full path.

This screenshot shows how modprobe loads the isdn module, automatically loading slhc in background.

```
[root@RHEL52 kernel]# lsmod | grep isdn
[root@RHEL52 kernel]# modprobe isdn
[root@RHEL52 kernel]# lsmod | grep isdn
isdn                  122433  0
slhc                   10561  1 isdn
[root@RHEL52 kernel]#
```

4.4.9. /lib/modules/<kernel>/modules.dep

Module dependencies are stored in **modules.dep**.

```
[root@RHEL52 2.6.18-92.1.18.el5]# pwd
/lib/modules/2.6.18-92.1.18.el5
[root@RHEL52 2.6.18-92.1.18.el5]# head -3 modules.dep
/lib/modules/2.6.18-92.1.18.el5/kernel/drivers/net/tokenring/3c359.ko:
/lib/modules/2.6.18-92.1.18.el5/kernel/drivers/net/pcmcia/3c574_cs.ko:
/lib/modules/2.6.18-92.1.18.el5/kernel/drivers/net/pcmcia/3c589_cs.ko:
```

4.4.10. depmod

The **modules.dep** file can be updated (recreated) with the **depmod** command. In this screenshot no modules were added, so **depmod** generates the same file.

```
root@barry:/lib/modules/2.6.17-2-686# ls -l modules.dep
-rw-r--r-- 1 root root 310676 2008-03-01 16:32 modules.dep
root@barry:/lib/modules/2.6.17-2-686# depmod
root@barry:/lib/modules/2.6.17-2-686# ls -l modules.dep
-rw-r--r-- 1 root root 310676 2008-12-07 13:54 modules.dep
```

4.4.11. rmmod

Similar to insmod, the **rmmod** command is rarely used anymore.

```
[root@RHELv4u3 ~]# modprobe isdn
[root@RHELv4u3 ~]# rmmod slhc
ERROR: Module slhc is in use by isdn
[root@RHELv4u3 ~]# rmmod isdn
[root@RHELv4u3 ~]# rmmod slhc
[root@RHELv4u3 ~]# lsmod | grep isdn
[root@RHELv4u3 ~]#
```

4.4.12. modprobe -r

Contrary to rmmod, **modprobe** will automatically remove unneeded modules.

```
[root@RHELv4u3 ~]# modprobe isdn
```

```
[root@RHELv4u3 ~]# lsmod | grep isdn
isdn                  133537  0
slhc                    7233  1 isdn
[root@RHELv4u3 ~]# modprobe -r isdn
[root@RHELv4u3 ~]# lsmod | grep isdn
[root@RHELv4u3 ~]# lsmod | grep slhc
[root@RHELv4u3 ~]#
```

4.4.13. /etc/modprobe.conf

The **/etc/modprobe.conf** file and the **/etc/modprobe.d** directory can contain aliases (used by humans) and options (for dependent modules) for modprobe.

```
[root@RHEL52 ~]# cat /etc/modprobe.conf
alias scsi_hostadapter mptbase
alias scsi_hostadapter1 mptspi
alias scsi_hostadapter2 ata_piix
alias eth0 pcnet32
alias eth2 pcnet32
alias eth1 pcnet32
```

4.5. compiling a kernel

4.5.1. extraversion

Enter into **/usr/src/redhat/BUILD/kernel-2.6.9/linux-2.6.9/** and change the **extraversion** in the Makefile.

```
[root@RHEL52 linux-2.6.18.i686]# pwd
/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i686
[root@RHEL52 linux-2.6.18.i686]# vi Makefile
[root@RHEL52 linux-2.6.18.i686]# head -4 Makefile
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 18
EXTRAVERSION = -paul2008
```

4.5.2. make mrproper

Now clean up the source from any previous installs with **make mrproper**. If this is your first after downloading the source code, then this is not needed.

```
[root@RHEL52 linux-2.6.18.i686]# make mrproper
  CLEAN   scripts/basic
  CLEAN   scripts/kconfig
  CLEAN   include/config
  CLEAN   .config .config.old
```

4.5.3. .config

Now copy a working **.config** from /boot to our kernel directory. This file contains the configuration that was used for your current working kernel. It determines whether modules are included in compilation or not.

```
[root@RHEL52 linux-2.6.18.i686]# cp /boot/config-2.6.18-92.1.18.el5 .config
```

4.5.4. make menuconfig

Now run **make menuconfig** (or the graphical **make xconfig**). This tool allows you to select whether to compile stuff as a module (m), as part of the kernel (*), or not at all (smaller kernel size). If you remove too much, your kernel will not work. The configuration will be stored in the hidden **.config** file.

```
[root@RHEL52 linux-2.6.18.i686]# make menuconfig
```

4.5.5. make clean

Issue a **make clean** to prepare the kernel for compile. **make clean** will remove most generated files, but keeps your kernel configuration. Running a **make mrproper** at this point would destroy the .config file that you built with **make menuconfig**.

```
[root@RHEL52 linux-2.6.18.i686]# make clean
```

4.5.6. make bzImage

And then run **make bzImage**, sit back and relax while the kernel compiles. You can use **time make bzImage** to know how long it takes to compile, so next time you can go for a short walk.

```
[root@RHEL52 linux-2.6.18.i686]# time make bzImage
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/basic/docproc
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/kxgettext.o
...
```

This command will end with telling you the location of the **bzImage** file (and with time info if you also specified the time command.

```
Kernel: arch/i386/boot/bzImage is ready  (#1)
```

```
real 13m59.573s
user 1m22.631s
sys 11m51.034s
[root@RHEL52 linux-2.6.18.i686]#
```

You can already copy this image to /boot with **cp arch/i386/boot/bzImage /boot/vmlinuz-<kernel-version>**.

4.5.7. make modules

Now run **make modules**. It can take 20 to 50 minutes to compile all the modules.

```
[root@RHEL52 linux-2.6.18.i686]# time make modules
  CHK     include/linux/version.h
  CHK     include/linux/utsrelease.h
  CC [M]  arch/i386/kernel/msr.o
  CC [M]  arch/i386/kernel/cpuid.o
  CC [M]  arch/i386/kernel/microcode.o
```

4.5.8. make modules_install

To copy all the compiled modules to **/lib/modules** just run **make modules_install** (takes about 20 seconds). Here's a screenshot from before the command.

```
[root@RHEL52 linux-2.6.18.i686]# ls -l /lib/modules/
total 20
drwxr-xr-x 6 root root 4096 Oct 15 13:09 2.6.18-92.1.13.el5
drwxr-xr-x 6 root root 4096 Nov 11 08:51 2.6.18-92.1.17.el5
drwxr-xr-x 6 root root 4096 Dec  6 07:11 2.6.18-92.1.18.el5
[root@RHEL52 linux-2.6.18.i686]# make modules_install
```

And here is the same directory after. Notice that **make modules_install** created a new directory for the new kernel.

```
[root@RHEL52 linux-2.6.18.i686]# ls -l /lib/modules/
total 24
drwxr-xr-x 6 root root 4096 Oct 15 13:09 2.6.18-92.1.13.el5
drwxr-xr-x 6 root root 4096 Nov 11 08:51 2.6.18-92.1.17.el5
drwxr-xr-x 6 root root 4096 Dec  6 07:11 2.6.18-92.1.18.el5
drwxr-xr-x 3 root root 4096 Dec  6 08:50 2.6.18-paul2008
```

4.5.9. /boot

We still need to copy the kernel, the System.map and our configuration file to /boot. Strictly speaking the .config file is not obligatory, but it might help you in future compilations of the kernel.

```
[root@RHEL52 ]# pwd
/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i686
[root@RHEL52 ]# cp System.map /boot/System.map-2.6.18-paul2008
[root@RHEL52 ]# cp .config /boot/config-2.6.18-paul2008
[root@RHEL52 ]# cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.18-paul2008
```

4.5.10. mkinitrd

The kernel often uses an initrd file at bootup. We can use **mkinitrd** to generate this file. Make sure you use the correct kernel name!

```
[root@RHEL52 ]# pwd
/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i686
[root@RHEL52 ]# mkinitrd /boot/initrd-2.6.18-paul2008 2.6.18-paul2008
```

4.5.11. bootloader

Compilation is now finished, don't forget to create an additional stanza in grub or lilo.

4.6. compiling one module

4.6.1. hello.c

A little C program that will be our module.

```
[root@rhel4a kernel_module]# cat hello.c
#include <linux/module.h>
#include <section>

int init_module(void)
{
 printk(KERN_INFO "Start Hello World...\n");
 return 0;
}

void cleanup_module(void)
{
 printk(KERN_INFO "End Hello World... \n");
}
```

4.6.2. Makefile

The make file for this module.

```
[root@rhel4a kernel_module]# cat Makefile
obj-m += hello.o
```

```
all:
make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

These are the only two files needed.

```
[root@rhel4a kernel_module]# ll
total 16
-rw-rw-r--  1 paul paul 250 Feb 15 19:14 hello.c
-rw-rw-r--  1 paul paul 153 Feb 15 19:15 Makefile
```

4.6.3. make

The running of the **make** command.

```
[root@rhel4a kernel_module]# make
make -C /lib/modules/2.6.9-paul-2/build M=~/kernel_module modules
make[1]: Entering dir... `/usr/src/redhat/BUILD/kernel-2.6.9/linux-2.6.9'
CC [M]  /home/paul/kernel_module/hello.o
Building modules, stage 2.
MODPOST
CC      /home/paul/kernel_module/hello.mod.o
LD [M]  /home/paul/kernel_module/hello.ko
make[1]: Leaving dir... `/usr/src/redhat/BUILD/kernel-2.6.9/linux-2.6.9'
[root@rhel4a kernel_module]#
```

Now we have more files.

```
[root@rhel4a kernel_module]# ll
total 172
-rw-rw-r--  1 paul paul   250 Feb 15 19:14 hello.c
-rw-r--r--  1 root root 64475 Feb 15 19:15 hello.ko
-rw-r--r--  1 root root   632 Feb 15 19:15 hello.mod.c
-rw-r--r--  1 root root 37036 Feb 15 19:15 hello.mod.o
-rw-r--r--  1 root root 28396 Feb 15 19:15 hello.o
-rw-rw-r--  1 paul paul   153 Feb 15 19:15 Makefile
[root@rhel4a kernel_module]#
```

4.6.4. hello.ko

Use **modinfo** to verify that it is really a module.

```
[root@rhel4a kernel_module]# modinfo hello.ko
filename:       hello.ko
vermagic:       2.6.9-paul-2 SMP 686 REGPARM 4KSTACKS gcc-3.4
depends:
[root@rhel4a kernel_module]#
```

Good, so now we can load our hello module.

```
[root@rhel4a kernel_module]# lsmod | grep hello
[root@rhel4a kernel_module]# insmod ./hello.ko
[root@rhel4a kernel_module]# lsmod | grep hello
hello                   5504  0
[root@rhel4a kernel_module]# tail -1 /var/log/messages
Feb 15 19:16:07 rhel4a kernel: Start Hello World...
[root@rhel4a kernel_module]# rmmod hello
[root@rhel4a kernel_module]#
```

Finally **/var/log/messages** has a little surprise.

```
[root@rhel4a kernel_module]# tail -2 /var/log/messages
Feb 15 19:16:07 rhel4a kernel: Start Hello World...
Feb 15 19:16:35 rhel4a kernel: End Hello World...
[root@rhel4a kernel_module]#
```

Chapter 5. Introduction to Networking

5.1. About TCP/IP

5.1.1. Overview of tcp/ip v4

The unicast **Internet Protocol** is one of the oldest network protocols, commonly used today for LAN and WAN networks. Every **host** gets a unique 32-bit **ip-address**, this is either static or received from a **DHCP** server. Internet networks contain several **subnets**. Those subnets used to be **classful** (A,B,C,D or E), but this wasted a lot of address space. Today we work with **CIDR** notation to determine **network id** and **host id**.

In a couple of years we will all be using IPv6! *At least, that is what people say since 1995...*

5.1.2. Internet and routers

The internet is a collection of **routers** that act as gateways between different **segments**. Routers use their **routing table** to determine the route of tcp/ip **packets**. Routers are **layer 3** devices, layer 2 contains **bridges** and **switches**, layer 1 is cabling with **repeaters** and **hubs**. Layer 2 devices know your 48-bit unique in the world **MAC** address.

5.1.3. many protocols

For reliable connections, you use **tcp**, whereas **udp** is connectionless but faster. The **icmp** error messages are used by **ping**, multicast groups are managed by **igmp** and the ip to mac resolution is done by the **broadcast** protocol **arp**.

These protocols are visible in the protocol field of the ip header, and are listed in the **/etc/protocols** file.

```
paul@laika:~$ grep tcp /etc/protocols
tcp     6       TCP             # transmission control protocol
paul@laika:~$
```

Every host receives a **hostname**, usually placed in a **DNS name space** forming the **FQDN** or Fully Qualified Domain Name. Common application level protocols like SMTP, HTTP, SSH, telnet and FTP have fixed **port numbers**.

To find a port number, look in **/etc/services**.

```
paul@laika:~$ grep tftp /etc/services
tftp            69/udp
paul@laika:~$
```

5.1.4. Practice TCP/IP

1. Which ports are used by http, pop3, ssh, telnet, nntp and ftp ?

2. Explain why e-mail and websites are sent over tcp, whereas internet streaming radio and live broadcasts are using udp.

5.2. Using TCP/IP

5.2.1. to GUI or not to GUI

If you can, setup your tcp/ip configuration at install time, otherwise use the graphical tool from your distribution. In the case of RHEL, this is the **Network Administration Tool**, Novell and OpenSUSE users can use YaST. Avoid mixed use of the GUI tool with command line or direct editing of network configuration files. You should choose only one method to manage these files, because many GUI tools will override your manually edited settings. Also, on Red Hat Servers avoid editing the files in **/etc/sysconfig/networking** manually!

Now that we settled this, let's take a look at the files and script that configure your network.

5.2.2. /sbin/ifconfig

You can use the **ifconfig** command to see the tcp/ip configuration of a network interface. The first ethernet network card on linux is eth0.

```
[root@RHEL4b ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:3B:15:80
          inet addr:192.168.1.191  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe3b:1580/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:84 errors:0 dropped:0 overruns:0 frame:0
          TX packets:80 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9216 (9.0 KiB)  TX bytes:8895 (8.6 KiB)
          Interrupt:185 Base address:0x1400

[root@RHEL4b ~]#
```

You can also disable a network interface with **ifconfig eth0 down**, or enable it with **ifconfig eth0 up**.

Every user has access to /sbin/ifconfig, providing the path is set. Normal users cannot use it to disable or enable interfaces, or set the ip address.

```
[root@RHEL4b ~]# ifconfig eth0 192.168.1.199
[root@RHEL4b ~]#
```

The ip address change will be valid until the next change, or until reboot. You can also supply the **subnet mask** with ifconfig.

```
root@laika:~# ifconfig eth0 192.168.1.40 netmask 255.255.255.0
root@laika:~#
```

Careful, if you try this via an ssh connection, then you might lose your ssh connection.

5.2.3. /etc/init.d/network(ing)

If you have a problem with network interfaces, you can try to restart the network init script, as shown here on Ubuntu 7.04. The script stops and starts the interfaces, and renews an ip configuration with the DHCP server.

```
root@laika:~# /etc/init.d/networking restart
 * Reconfiguring network interfaces...
There is already a pid file /var/run/dhclient.eth0.pid with pid 14570
killed old client process, removed PID file
Internet Systems Consortium DHCP Client V3.0.4
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:90:f5:4e:ae:17
Sending on   LPF/eth0/00:90:f5:4e:ae:17
Sending on   Socket/fallback
DHCPRELEASE on eth0 to 192.168.1.1 port 67
There is already a pid file /var/run/dhclient.eth0.pid with pid 134993416
Internet Systems Consortium DHCP Client V3.0.4
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:90:f5:4e:ae:17
Sending on   LPF/eth0/00:90:f5:4e:ae:17
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 5
DHCPOFFER from 192.168.1.1
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.1
bound to 192.168.1.40 -- renewal in 249143 seconds.
root@laika:~#
```

5.2.4. /etc/sysconfig

Red Hat derived Linux systems store their network configuration files in the **/etc/sysconfig/** directory. Debian derived systems do not have this directory.

5.2.4.1. /etc/sysconfig/network

Routing and host information for all network interfaces is specified in the **/etc/sysconfig/network** file. Below an example, setting 192.168.1.1 as the router (default

gateway), and leaving the default hostname of localhost.localdomain. Common options not shown in this screenshot are **GATEWAYDEV** to set one of your network cards as the gateway device, and **NISDOMAIN** to specify the NIS domain name.

```
paul@RHELv4u2:~$ cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=localhost.localdomain
GATEWAY=192.168.1.1
```

The same file, but here the hostname of the machine is not set to the default as above.

```
[paul@RHEL4b ~]$ cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=RHEL4b
[paul@RHEL4b ~]$
```

5.2.4.2. /etc/sysconfig/network-scripts

For every network card in your computer, you should have an interface configuration file named **/etc/sysconfig/network-scripts/ifcfg-$IFNAME**. Be careful when editing these files, your edits will work, until you start the **system-config-network** (might soon be renamed to redhat-config-network) tool. This tool can and will overwrite your manual edits.

The first ethernet NIC will get **ifcfg-eth0**, the next one ifcfg-eth1 and so on. Below is an example.

```
paul@RHELv4u2:~$ cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.1.255
HWADDR=00:0C:29:5A:86:D7
IPADDR=192.168.1.222
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
TYPE=Ethernet
```

When the second nic is configured for dhcp, then this is the ifcfg-eth1.

```
paul@RHELv4u2:~$ cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BOOTPROTO=dhcp
HWADDR=00:0C:29:6A:34:D8
ONBOOT=yes
TYPE=Ethernet
```

Besides **dhcp** and **bootp** the BOOTPROTO variable can be **static** or **none**, both meaning there should be no protocol used at boottime to set the interface values. The BROADCAST variable is no longer needed, it will be calculated.

The HWADDR can be used to make sure that the nic's get the correct name when multiple nic's are present in the computer. It can not be used to set the MAC address of a nic. For this, you need to specify the MACADDR variable. Do not use HWADDR and MACADDR in the same ifcfg file.

5.2.5. /sbin/ifup and /sbin/ifdown

The **ifup** and **ifdown** commands take an interface as argument and bring it up or down. The screenshot below deactivates the eth0 network interface.

```
root@laika:~# ifdown eth0
There is already a pid file /var/run/dhclient.eth0.pid with pid 14925
killed old client process, removed PID file
Internet Systems Consortium DHCP Client V3.0.4
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:90:f5:4e:ae:17
Sending on   LPF/eth0/00:90:f5:4e:ae:17
Sending on   Socket/fallback
DHCPRELEASE on eth0 to 192.168.1.1 port 67
```

On debian derived systems, these commands will look at **/etc/network/interfaces**, whereas on Red Hat derived systems they will look at /etc/sysconfig/network-scripts/ifcfg- files. In the screenshot below ifup is used to bring up the eth0 interface. Because the /etc/network/interfaces file says eth0 uses DHCP, the ifup tool will (try to) start the dhclient daemon.

```
root@laika:~# ifup eth0
There is already a pid file /var/run/dhclient.eth0.pid with pid 134993416
Internet Systems Consortium DHCP Client V3.0.4
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:90:f5:4e:ae:17
Sending on   LPF/eth0/00:90:f5:4e:ae:17
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 8
DHCPOFFER from 192.168.1.1
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.1
bound to 192.168.1.40 -- renewal in 231552 seconds.
root@laika:~#
```

5.2.6. /sbin/dhclient

Home and client Linux desktops often have **dhclient** running. This is a daemon that enables a network interface to lease an ip configuration from a DHCP server. When your adapter is configured for DHCP or BOOTP, then /sbin/ifup will start the dhclient daemon.

5.2.7. /sbin/route

You can see the computer's local routing table with the **route** command (and also with **netstat -r**).

```
root@RHEL4b ~]# netstat -r
Kernel IP routing table
Destination     Gateway   Genmask         Flags   MSS Window  irtt Iface
192.168.1.0     *         255.255.255.0   U         0 0          0 eth0
[root@RHEL4b ~]# route
Kernel IP routing table
Destination     Gateway   Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *         255.255.255.0   U     0      0        0 eth0
[root@RHEL4b ~]#
```

It appears this computer does not have a **gateway** configured, so we use **route add default gw** to add a **default gateway**.

```
[root@RHEL4b ~]# route add default gw 192.168.1.1
[root@RHEL4b ~]# route
Kernel IP routing table
Destination     Gateway      Genmask        Flags Metric Ref  Use Iface
192.168.1.0     *            255.255.255.0  U     0      0      0 eth0
default         192.168.1.1  0.0.0.0        UG    0      0      0 eth0
[root@RHEL4b ~]#
```

5.2.8. arp

Mac to IP resolution is handled by the **arp** protocol. The arp table can be displayed with the arp tool.

```
root@barry:~# arp -a
? (192.168.1.191) at 00:0C:29:3B:15:80 [ether] on eth1
agapi (192.168.1.73) at 00:03:BA:09:7F:D2 [ether] on eth1
anya (192.168.1.1) at 00:12:01:E2:87:FB [ether] on eth1
faith (192.168.1.41) at 00:0E:7F:41:0D:EB [ether] on eth1
kiss (192.168.1.49) at 00:D0:E0:91:79:95 [ether] on eth1
laika (192.168.1.40) at 00:90:F5:4E:AE:17 [ether] on eth1
pasha (192.168.1.71) at 00:03:BA:02:C3:82 [ether] on eth1
shaka (192.168.1.72) at 00:03:BA:09:7C:F9 [ether] on eth1
root@barry:~#
```

Anya is a Cisco Firewall, Faith is an HP Color printer, Kiss is a Kiss DP600, laika is a Clevo laptop and Agapi, Shaka and Pasha are SPARC servers. The question mark is a Red Hat Enterprise Linux server running in vmware.

5.2.9. ping

If you can ping to another host, then ip is configured.

```
[root@RHEL4b ~]# ping 192.168.1.5
PING 192.168.1.5 (192.168.1.5) 56(84) bytes of data.
64 bytes from 192.168.1.5: icmp_seq=0 ttl=64 time=1004 ms
64 bytes from 192.168.1.5: icmp_seq=1 ttl=64 time=1.19 ms
64 bytes from 192.168.1.5: icmp_seq=2 ttl=64 time=0.494 ms
64 bytes from 192.168.1.5: icmp_seq=3 ttl=64 time=0.419 ms

--- 192.168.1.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3009ms
rtt min/avg/max/mdev = 0.419/251.574/1004.186/434.520 ms, pipe 2
[root@RHEL4b ~]#
```

5.2.10. Red Hat network settings backup

It is always a good idea to have a backup of current network settings. The **system-config-network-cmd** can do this for you.

```
root ~# system-config-network-cmd -e > NetworkSettings20070208.txt
```

And system-config-network-cmd can also be used to restore these settings.

```
root ~# system-config-network-cmd -i -c < NetworkSettings20070208.txt
```

For other Linux Systems, take a backup of the relevant portions in /etc.

5.2.11. Restarting the network

To stop, start or restart all network interfaces and services, use **service network stop|start|restart**. *Do not stop the network when connected through ssh.*

5.2.12. ethtool

To display or change network card settings, use **ethtool**. The results depend on the capabilities of your network card. The example shows a network that auto-negotiates it's bandwidth.

```
root@laika:~# ethtool eth0
Settings for eth0:
 Supported ports: [ TP ]
 Supported link modes:   10baseT/Half 10baseT/Full
                         100baseT/Half 100baseT/Full
                         1000baseT/Full
 Supports auto-negotiation: Yes
 Advertised link modes:  10baseT/Half 10baseT/Full
                         100baseT/Half 100baseT/Full
                         1000baseT/Full
 Advertised auto-negotiation: Yes
 Speed: 1000Mb/s
 Duplex: Full
 Port: Twisted Pair
 PHYAD: 0
```

```
 Transceiver: internal
 Auto-negotiation: on
 Supports Wake-on: pumbg
 Wake-on: g
 Current message level: 0x00000033 (51)
 Link detected: yes
```

This example shows how to use ethtool to switch the bandwidth from 1000Mbit to 100Mbit and back. Note that some time passes before the nic is back to 1000Mbit.

```
root@laika:~# ethtool eth0 | grep Speed
 Speed: 1000Mb/s
root@laika:~# ethtool -s eth0 speed 100
root@laika:~# ethtool eth0 | grep Speed
 Speed: 100Mb/s
root@laika:~# ethtool -s eth0 speed 1000
root@laika:~# ethtool eth0 | grep Speed
 Speed: 1000Mb/s
```

5.2.13. Practice IP Configuration

1. Use ifconfig to list all your network interfaces and their ip-addresses. Write down your ip-address and subnet mask.

2. Use the GUI tool of your distro to set a fix ip address (use the same address as the one you got from dhcp). Verify with ifconfig and ping to a neighbour that it works. Also look at the configuration files in /etc/network or /etc/sysconfig to see how the GUI tool sets a fixed address.

3. Use the GUI tool to enable dhcp again (and verify the changes in the config files).

4. Use ifdown or ifconfig to disable your eth0 network card.

5. Restart networking to enable your network card again.

6. Is the dhclient daemon running ?

7. Verify that you have a default gateway.

8. Ping the default gateway, then look at the MAC address of the default gateway.

5.3. multiple IP adresses

5.3.1. Binding multiple ip-addresses

To bind more than one ip-addres to the same interface, use **ifcfg-eth0:0**, where the last zero can be anything else. Only two directives are required in the file.

```
root@RHELv4u2:/etc/sysconfig/network-scripts# cat ifcfg-eth0:0
DEVICE=eth0:0
```

```
IPADDR=192.168.1.232
```

5.3.2. Enabling extra ip-addresses

To activate a virtual network interface, use **ifup**, to deactivate it, use **ifdown**.

```
root@RHELv4u2:~# ifdown eth0:0
root@RHELv4u2:~# ifup eth0:0
```

5.3.3. Practice multiple IP addresses

1. Add an extra ip address to your server. Test that it works (have your neighbour ssh to it)!

2. Use ifdown and ifup to disable and enable the second ip address.

5.4. multihomed hosts

5.4.1. bonding

You can combine (bond) two physical network interfaces as one logical interface. Having two network cards serve the same IP-address doubles the bandwidth, and provides hardware redundancy. For **bonding** to work, you have to load the kernel module for bonding. You can do this manually with **modprobe**.

```
root@RHELv4u2:~# modprobe bonding
root@RHELv4u2:~# lsmod | grep bon
bonding               58984  0
```

Or automatically, by adding the alias to **/etc/modprobe.conf** (used to be called /etc/modules.conf).

```
root@RHELv4u2:~# echo alias bond0 bonding >> /etc/modprobe.conf
```

You need two network cards to enable bonding, and add the **MASTER** and **SLAVE** variables. In this case we used eth0 and eth1, configured like this.

```
root@RHELv4u2:~# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BROADCAST=192.168.1.255
HWADDR=00:0C:29:5A:86:D7
IPADDR=192.168.1.222
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
```

```
TYPE=Ethernet
GATEWAY=192.168.1.1
MASTER=bond0
SLAVE=yes
USERCTL=no
root@RHELv4u2:~# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
BROADCAST=192.168.1.255
HWADDR=00:0C:29:5A:86:E1
IPADDR=192.168.1.232
NETMASK=255.255.255.0
NETWORK=192.168.1.0
ONBOOT=yes
TYPE=Ethernet
GATEWAY=192.168.1.1
MASTER=bond0
SLAVE=yes
USERCTL=no
root@RHELv4u2:~#
```

And you need to set up a bonding interface. In this case, we call it bond0.

```
root@RHELv4u2:~# cat /etc/sysconfig/network-scripts/ifcfg-bond0
DEVICE=bond0
BOOTPROTO=none
ONBOOT=no
NETWORK=192.168.1.0
NETMASK=255.255.255.0
IPADDR=192.168.1.229
USERCTL=no
root@RHELv4u2:~#
```

To bring up the interface, just use the **ifup bond0** command.

```
root@RHELv4u2:/etc/sysconfig/network-scripts# ifup bond0
Enslaving eth0 to bond0
Enslaving eth1 to bond0
root@RHELv4u2:~#
```

The **ifconfig** command will show you all activated interfaces.

```
root@RHELv4u2:~# ifconfig
bond0     Link encap:Ethernet  HWaddr 00:0C:29:5A:86:D7
inet addr:192.168.1.229  Bcast:192.168.1.255  Mask:255.255.255.0
inet6 addr: fe80::200:ff:fe00:0/64 Scope:Link
UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
RX packets:3835 errors:0 dropped:0 overruns:0 frame:0
TX packets:1001 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:469645 (458.6 KiB)  TX bytes:139816 (136.5 KiB)

eth0      Link encap:Ethernet  HWaddr 00:0C:29:5A:86:D7
inet6 addr: fe80::20c:29ff:fe5a:86d7/64 Scope:Link
UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
RX packets:3452 errors:0 dropped:0 overruns:0 frame:0
TX packets:837 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:412155 (402.4 KiB)  TX bytes:117844 (115.0 KiB)
```

```
Interrupt:11 Base address:0x1400

eth1      Link encap:Ethernet  HWaddr 00:0C:29:5A:86:D7
inet6 addr: fe80::20c:29ff:fe5a:86d7/64 Scope:Link
UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
RX packets:392 errors:0 dropped:0 overruns:0 frame:0
TX packets:177 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:58084 (56.7 KiB)  TX bytes:24078 (23.5 KiB)
Interrupt:10 Base address:0x1480
```

5.4.2. /proc/net/bond*

You can verify the proper working of the bonding interfaces by looking at **/proc/net/bonding/**. Below is a screenshot of a Red Hat Enterprise 5 server, with eth1 and eth2 in bonding.

```
[root@RHEL5 ~]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.1.2 (January 20, 2007)

Bonding Mode: load balancing (round-robin)
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:0c:29:a0:9d:e3

Slave Interface: eth2
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:0c:29:a0:9d:ed
[root@RHEL5 ~]#
```

5.4.3. Practice multihomed hosts

1. Add a network card to the vmware machine, and bond the two cards as one virtual (double bandwidth and failover) card.

5.5. Introduction to iptables

5.5.1. Introducing iptables

The Linux kernel has a built-in stateful firewall named iptables. To stop the **iptables** firewall on Red Hat, use the service command.

```
root@RHELv4u4:~# service iptables stop
```

```
Flushing firewall rules:                                   [  OK  ]
Setting chains to policy ACCEPT: filter                    [  OK  ]
Unloading iptables modules:                                [  OK  ]
root@RHELv4u4:~#
```

The easy way to configure iptables, is to use a graphical tool like KDE's **kmyfirewall** or **Security Level Configuration Tool**. You can find the latter in the GUI menu, somewhere in System Tools - Security, or you can start it by typing **system-config-securitylevel** in bash. These tools allow for some basic firewall configuration. You can decide whether to enable or disable the firewall, and what typical standard ports are allowed when the firewall is active. You can even add some custom ports. When you are done, the configuration is written to **/etc/sysconfig/iptables** on Red Hat.

```
root@RHELv4u4:~# cat /etc/sysconfig/iptables
# Firewall configuration written by system-config-securitylevel
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-F...NPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A RH-F...NPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A RH-F...NPUT -m state --state NEW -m tcp -p tcp --dport 21 -j ACCEPT
-A RH-F...NPUT -m state --state NEW -m tcp -p tcp --dport 25 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
root@RHELv4u4:~#
```

To start the service, issue the **service iptables start** command. You can configure iptables to start at boot time with chkconfig.

```
root@RHELv4u4:~# service iptables start
Applying iptables firewall rules:                          [  OK  ]
root@RHELv4u4:~# chkconfig iptables on
root@RHELv4u4:~#
```

One of the nice features of iptables is that it displays extensive **status** information when queried with the **service iptables status** command.

```
root@RHELv4u4:~# service iptables status
Table: filter
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
RH-Firewall-1-INPUT  all  --  0.0.0.0/0           0.0.0.0/0
```

```
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
RH-Firewall-1-INPUT  all  --  0.0.0.0/0           0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain RH-Firewall-1-INPUT (2 references)
target  prot opt source      destination
ACCEPT  all  --  0.0.0.0/0   0.0.0.0/0
ACCEPT  icmp --  0.0.0.0/0   0.0.0.0/0   icmp type 255
ACCEPT  esp  --  0.0.0.0/0   0.0.0.0/0
ACCEPT  ah   --  0.0.0.0/0   0.0.0.0/0
ACCEPT  udp  --  0.0.0.0/0   224.0.0.251 udp dpt:5353
ACCEPT  udp  --  0.0.0.0/0   0.0.0.0/0   udp dpt:631
ACCEPT  all  --  0.0.0.0/0   0.0.0.0/0   state RELATED,ESTABLISHED
ACCEPT  tcp  --  0.0.0.0/0   0.0.0.0/0   state NEW tcp dpt:22
ACCEPT  tcp  --  0.0.0.0/0   0.0.0.0/0   state NEW tcp dpt:80
ACCEPT  tcp  --  0.0.0.0/0   0.0.0.0/0   state NEW tcp dpt:21
ACCEPT  tcp  --  0.0.0.0/0   0.0.0.0/0   state NEW tcp dpt:25
REJECT  all  --  0.0.0.0/0   0.0.0.0/0   reject-with icmp-host-prohibited

root@RHELv4u4:~#
```

Mastering firewall configuration requires a decent knowledge of tcp/ip. Good iptables tutorials can be found online here http://iptables-tutorial.frozentux.net/iptables-tutorial.html and here http://tldp.org/HOWTO/IP-Masquerade-HOWTO/.

5.5.2. Practice iptables

1. Verify whether the firewall is running.

2. Disable the firewall.

5.6. xinetd and inetd

5.6.1. About the superdaemon

Back when resources like RAM memory were limited, a super-server was deviced to listen to all sockets and start the appropriate daemon only when needed. Services like swat, telnet and vmware are typically served by such a super-server. The **xinetd** superserver is more recent than **inetd**. We will discuss the configuration both daemons.

Recent Linux distributions like RHEL5 and Ubuntu8.04 do not install inetd or xinetd by default.

5.6.2. inetd or xinetd

First verify whether your computer is running inetd or xinetd. This Debian 4.0 Etch is running inetd.

```
root@barry:~# ps fax | grep inet
 3870 ?        Ss     0:00 /usr/sbin/inetd
```

This Red Hat Enterprise Linux 4 update 4 is running xinetd.

```
[root@RHEL4b ~]# ps fax | grep inet
 3003 ?        Ss     0:00 xinetd -stayalive -pidfile /var/run/xinetd.pid
```

Both daemons have the same functionality (listening to many ports, starting other daemons when they are needed), but they have different configuration files.

5.6.3. The superdaemon xinetd

The **xinetd** daemon is often called a superdaemon because it listens to a lot of incoming connections, and starts other daemons when they are needed. When a connection request is received, xinetd will first check TCP wrappers (/etc/hosts.allow and /etc/hosts.deny) and then give control of the connection to the other daemon. This superdaemon is configured through **/etc/xinetd.conf** and the files in the directory **/etc/xinetd.d**. Let's first take a look at /etc/xinetd.conf.

```
paul@RHELv4u2:~$ cat /etc/xinetd.conf
#
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/

defaults
{
instances               = 60
log_type                = SYSLOG authpriv
log_on_success          = HOST PID
log_on_failure          = HOST
cps                     = 25 30
}

includedir /etc/xinetd.d

paul@RHELv4u2:~$
```

According to the settings in this file, xinetd can handle 60 client requests at once. It uses the **authpriv** facility to log the host ip-address and pid of successful daemon spawns. When a service (aka protocol linked to daemon) gets more than 25 cps (connections per second), it holds subsequent requests for 30 seconds.

The directory **/etc/xinetd.d** contains more specific configuration files. Let's also take a look at one of them.

```
paul@RHELv4u2:~$ ls /etc/xinetd.d
amanda     chargen-udp  echo      klogin       rexec   talk
```

```
amandaidx  cups-lpd     echo-udp  krb5-telnet  rlogin  telnet
amidxtape  daytime      eklogin   kshell       rsh     tftp
auth       daytime-udp  finger    ktalk        rsync   time
chargen    dbskkd-cdb   gssftp    ntalk        swat    time-udp
paul@RHELv4u2:~$ cat /etc/xinetd.d/swat
# default: off
# description: SWAT is the Samba Web Admin Tool. Use swat \
#              to configure your Samba server. To use SWAT, \
#              connect to port 901 with your favorite web browser.
service swat
{
port            = 901
socket_type     = stream
wait            = no
only_from       = 127.0.0.1
user            = root
server          = /usr/sbin/swat
log_on_failure  += USERID
disable         = yes
}
paul@RHELv4u2:~$
```

The services should be listed in the **/etc/services** file. Port determines the service port, and must be the same as the port specified in /etc/services. The **socket_type** should be set to **stream** for tcp services (and to dgram for udp). The **log_on_failure +=** concats the userid to the log message formatted in /etc/xinetd.conf. The last setting **disable** can be set to yes or no. Setting this to **no** means the service is enabled!

Check the xinetd and xinetd.conf manual pages for many more configuration options.

5.6.4. The superdaemon inetd

This superdaemon has only one configuration file **/etc/inetd.conf**. Every protocol or daemon that it is listening for, gets one line in this file.

```
root@barry:~# grep ftp /etc/inetd.conf
tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd /boot/tftp
root@barry:~#
```

You can disable a service in inetd.conf above by putting a # at the start of that line. Here an example of the disabled vmware web interface (listening on tcp port 902).

```
paul@laika:~$ grep vmware /etc/inetd.conf
#902 stream tcp nowait root /usr/sbin/vmware-authd vmware-authd
```

5.6.5. Practice

1. Verify on all systems whether they are using xinetd or inetd.

2. Look at the configuration files.

3. (If telnet is installable, then replace swat in these questions with telnet) Is swat installed ? If not, then install swat and look at the changes in the (x)inetd configuration. Is swat enabled or disabled ?

4. Disable swat, test it. Enable swat, test it.

5.7. OpenSSH

5.7.1. Secure Shell

Avoid using **telnet**, **rlogin** and **rsh** to remotely connect to your servers. These older protocols do not encrypt the login session, which means your user id and password can be sniffed by tools like **ethereal** aka wireshark. To securely connect to your servers, use **OpenSSH**. An ssh connection always starts with a cryptographic handshake, followed by encryption of the transport layer using a symmetric cypher. Then authentication takes place (using user id/password or public/private keys) and communication can take place over the encrypted connection. In other words, the tunnel is encrypted before you start typing anything.

The OpenSSH package is maintained by the **OpenBSD** people and is distributed with a lot of operating systems (it may even be the most popular package in the world). Below sample use of **ssh** to connect from one server (RHELv4u2) to another one (RHELv4u4).

```
paul@RHELv4u2:~$ ssh 192.168.1.220
The authenticity of host '192.168.1.220' can't be established.
RSA key fingerprint is c4:3c:52:e6:d8:8b:ce:17:8b:c9:78:5a:f3:51:06:4f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.220' (RSA) to the list of known...
paul@192.168.1.220's password:
Last login: Sun Jan 21 07:16:26 2007 from 192.168.1.40
paul@RHELv4u4:~$
```

The second time ssh remembers the connection. It added an entry to the ~/.ssh/known_hosts file.

```
paul@RHELv4u2:~$ ssh 192.168.1.220
paul@192.168.1.220's password:
Last login: Sun Jan 21 08:49:19 2007 from 192.168.1.222
paul@RHELv4u4:~$
```

5.7.2. SSH Protocol versions

The ssh protocol has two versions (1 and 2). Avoid using version 1 anywhere, since it contains some known vulnerabilities. You can control the protocol version via **/etc/ssh/ssh_config** for the client side and **/etc/ssh/sshd_config** for the openssh-server daemon.

```
root@laika:/etc/ssh# grep Protocol ssh_config
#   Protocol 2,1
root@laika:/etc/ssh# grep Protocol sshd_config
Protocol 2
root@laika:/etc/ssh#
```

Configuration of ssh is done in the **/etc/ssh** directory and is pretty straightforward.

5.7.3. About Public and Private keys

Imagine Alice and Bob, two people that like to communicate with eachother. Using public and private keys they can communicate with encryption and with authentication.

When Alice wants to send an encrypted message to Bob, she uses the public key of Bob. Bob shares his Public Key with Alice, but keeps his Private Key private! Since Bob is the only one to have Bob's Private Key, Alice is sure that Bob is the only one that can read the encrypted message.

When Bob wants to verify that the message came from Alice, Bob uses the Public Key of Alice to verify that Alice signed the message with her Private Key. Since Alice is the only one to have Alice's Private Key, Bob is sure the message came from Alice.

5.7.4. Setting up passwordless ssh

To set up passwordless ssh authentication through public/private keys, use **ssh-keygen** to generate a key pair without a passphrase, and then copy your public key to the destination server. Let's do this step by step.

In the example that follows, we will set up ssh without password between Alice and Bob. Alice has an account on a Red Hat Enterprise Linux server, Bob is using Ubuntu on his laptop. Bob wants to give Alice access using ssh and the public and private key system. This means that even if Bob changes his password on his laptop, Alice will still have access.

5.7.4.1. ssh-keygen

The example below shows how Alice uses **ssh-keygen** to generate a key pair. Alice does not enter a passphrase.

```
[alice@RHEL5 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/alice/.ssh/id_rsa):
Created directory '/home/alice/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alice/.ssh/id_rsa.
```

```
Your public key has been saved in /home/alice/.ssh/id_rsa.pub.
The key fingerprint is:
9b:ac:ac:56:c2:98:e5:d9:18:c4:2a:51:72:bb:45:eb alice@RHEL5
[alice@RHEL5 ~]$
```

5.7.4.2. ~/.ssh

While ssh-keygen generates a public and a private key, it will also create a hidden .ssh directory with proper permissions. If you create the .ssh directory manually, then you need to chmod 700 it! Otherwise ssh will refuse to use the keys (world readable private keys are not secure!).

As you can see, the .ssh directory is secure in Alice's home directory.

```
[alice@RHEL5 ~]$ ls -ld .ssh
drwx------ 2 alice alice 4096 May  1 07:38 .ssh
[alice@RHEL5 ~]$
```

Bob is using Ubuntu at home. He decides to manually create the .ssh directory, so he needs to manually secure it.

```
bob@laika:~$ mkdir .ssh
bob@laika:~$ ls -ld .ssh
drwxr-xr-x 2 bob bob 4096 2008-05-14 16:53 .ssh
bob@laika:~$ chmod 700 .ssh/
bob@laika:~$
```

5.7.4.3. id_rsa and id_rsa.pub

The ssh-keygen command generate two keys in .ssh. The public key is named **~/.ssh/id_rsa.pub**. The private key is named **~/.ssh/id_rsa**.

```
[alice@RHEL5 ~]$ ls -l .ssh/
total 16
-rw------- 1 alice alice 1671 May  1 07:38 id_rsa
-rw-r--r-- 1 alice alice  393 May  1 07:38 id_rsa.pub
[alice@RHEL5 ~]$
```

5.7.4.4. scp

To copy the public key from Alice's server tot Bob's laptop, Alice decides to use **scp**.

```
[alice@RHEL5 .ssh]$ scp id_rsa.pub bob@192.168.48.92:~/.ssh/authorized_keys
bob@192.168.48.92's password:
id_rsa.pub                                    100%  393     0.4KB/s   00:00
[alice@RHEL5 .ssh]$
```

Be careful when copying a second key! Do not overwrite the first key, instead append the key to the same ~/.ssh/authorized_keys file!

5.7.4.5. authorized_keys

In your ~/.ssh directory, you can create a file called **authorized_keys**. This file can contain one or more public keys from people you trust. Those trusted people can use their private keys to prove their identity and gain access to your account via ssh (without password). The example shows Bob's authorized_keys file containing the public key of Alice.

```
bob@laika:~$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEApCQ9xzyLzJes1sR+hPyqW2vyzt1D4zTLqk\
MDWBR4mMFuUZD/O583I3Lg/Q+JIq0RSksNzaL/BNLDou1jMpBe2Dmf/u22u4KmqlJBfDhe\
yTmGSBzeNYCYRSMq78CT9l9a+y6x/shucwhaILsy8A2XfJ9VCggkVtu7XlWFDL2cum08/0\
mRFwVrfc/uPsAn5XkkTscl4g21mQbnp9wJC40pGSJXXMuFOk8MgCb5ieSnpKFniAKM+tEo\
/vjDGSi3F/bxu691jscrU0VUdIoOSo98HUfEf7jKBRikxGAC7I4HLa+/zX73OIvRFAb2hv\
tUhn6RHrBtUJUjbSGiYeFTLDfcTQ== alice@RHEL5
bob@laika:~$
```

5.7.4.6. passwordless ssh

Alice can now use ssh to connect passwordless to Bob's laptop. In combination with ssh's capability to execute commands on the remote host, this can be useful in pipes across different machines.

```
[alice@RHEL5 ~]$ ssh bob@192.168.48.92 "ls -l .ssh"
total 4
-rw-r--r-- 1 bob bob 393 2008-05-14 17:03 authorized_keys
[alice@RHEL5 ~]$
```

5.7.5. X forwarding via SSH

The **ssh protocol** will remember the servers it connected to (and warn you in case something suspicious happened), and will use strong 128-bit encryption. Another popular feature of ssh is called **X11 forwarding** and is implemented with **ssh -X**.

Below an example of X11 forwarding: user paul logs in as user greet on her computer to start the graphical application mozilla-thunderbird. Although the application will run on the remote computer from greet, it will be displayed on the screen attached locally to paul's computer.

```
paul@laika:~/PDF$ ssh -X greet@greet.dyndns.org -p 55555
Warning: Permanently added the RSA host key for IP address \
'81.240.174.161' to the list of known hosts.
Password:
Linux raika 2.6.8-2-686 #1 Tue Aug 16 13:22:48 UTC 2005 i686 GNU/Linux
```

```
Last login: Thu Jan 18 12:35:56 2007
greet@raika:~$ ps fax | grep thun
greet@raika:~$ mozilla-thunderbird &
[1] 30336
```

5.7.6. Troubleshooting ssh

Use **ssh -v** to get debug information about the ssh connection attempt.

```
paul@laika:~$ ssh -v bert@192.168.1.192
OpenSSH_4.3p2 Debian-8ubuntu1, OpenSSL 0.9.8c 05 Sep 2006
debug1: Reading configuration data /home/paul/.ssh/config
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to 192.168.1.192 [192.168.1.192] port 22.
debug1: Connection established.
debug1: identity file /home/paul/.ssh/identity type -1
debug1: identity file /home/paul/.ssh/id_rsa type 1
debug1: identity file /home/paul/.ssh/id_dsa type -1
debug1: Remote protocol version 1.99, remote software version OpenSSH_3
debug1: match: OpenSSH_3.9p1 pat OpenSSH_3.*
debug1: Enabling compatibility mode for protocol 2.0
...
```

5.7.7. Practice SSH

1. Create a user for your neighbour, then test ssh to your neighbour (by ip-address or by hostname). (You might need to install the openssh-server with aptitude.)

2. Create a bookmark in Firefox, then close your firefox! Use ssh -X to run firefox on your screen, but on your neighbour's computer. Do you see your neighbour's bookmark ?

3. Verify in the ssh configuration files that only protocol version 2 is allowed.

4. Use ssh-keygen to create a keypair without passphrase. Setup passwordless ssh between you and your neighbour. (or between the ubuntu and the Red Hat)

5.8. Network File System

5.8.1. Network Attached Storage (NAS)

NAS means using separate servers with lots of storage, connected over a (hopefully very fast) network. NAS servers offer **file-based access** over the network with protocols like **NCP** (old Novell Netware), Sun's **NFS** (common on Unix) or **SMB** (implemented on Unix/Linux with Samba). NAS is not to be confused with **SAN**, which uses **block-based access** over proprietary protocols (Fiber Channel, iSCSI, ...).

A **NAS head** is a NAS without on-board storage, which connects to a SAN and acts as a translator between the file-level NAS protocols and the block-level SAN protocols.

5.8.2. NFS: the Network File System

5.8.2.1. protocol versions

The older **NFS** versions 2 and 3 are stateless (udp) by default, but they can use tcp. Clients connect to the server using **RPC** (on Linux this is controlled by the **portmap** daemon. Look at **rpcinfo** to verify that NFS and its related services are running.

```
root@RHELv4u2:~# /etc/init.d/portmap status
portmap (pid 1920) is running...
root@RHELv4u2:~# rpcinfo -p
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100024    1   udp  32768  status
100024    1   tcp  32769  status
root@RHELv4u2:~# service nfs start
Starting NFS services:                                     [  OK  ]
Starting NFS quotas:                                       [  OK  ]
Starting NFS daemon:                                       [  OK  ]
Starting NFS mountd:                                       [  OK  ]
```

The same rpcinfo command when NFS is started.

```
root@RHELv4u2:~# rpcinfo -p
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100024    1   udp  32768  status
100024    1   tcp  32769  status
100011    1   udp    985  rquotad
100011    2   udp    985  rquotad
100011    1   tcp    988  rquotad
100011    2   tcp    988  rquotad
100003    2   udp   2049  nfs
100003    3   udp   2049  nfs
100003    4   udp   2049  nfs
100003    2   tcp   2049  nfs
100003    3   tcp   2049  nfs
100003    4   tcp   2049  nfs
100021    1   udp  32770  nlockmgr
100021    3   udp  32770  nlockmgr
100021    4   udp  32770  nlockmgr
100021    1   tcp  32789  nlockmgr
100021    3   tcp  32789  nlockmgr
100021    4   tcp  32789  nlockmgr
100005    1   udp   1004  mountd
100005    1   tcp   1007  mountd
100005    2   udp   1004  mountd
100005    2   tcp   1007  mountd
100005    3   udp   1004  mountd
100005    3   tcp   1007  mountd
root@RHELv4u2:~#
```

NFS version 4 requires tcp (port 2049) and supports **Kerberos** user authentication as an option. NFS authentication only takes place when mounting the share. NFS versions 2 and 3 authenticate only the host.

5.8.2.2. server configuration

NFS is configured in **/etc/exports**. Here is a sample /etc/exports to explain the syntax. You need some way (NIS domain or LDAP) to synchronize userid's across computers when using NFS a lot. The **rootsquash** option will change UID 0 to the UID of the nfsnobody user account. The **sync** option will write writes to disk before completing the client request.

```
paul@laika:~$ cat /etc/exports
# Everyone can read this share
/mnt/data/iso  *(ro)

# Only the computers barry and pasha can readwrite this one
/var/www pasha(rw) barry(rw)

# same, but without root squashing for barry
/var/ftp pasha(rw) barry(rw,no_root_squash)

# everyone from the netsec.lan domain gets access
/var/backup       *.netsec.lan(rw)

# ro for one network, rw for the other
/var/upload   192.168.1.0/24(ro) 192.168.5.0/24(rw)
```

You don't need to restart the nfs server to start exporting your newly created exports. You can use the **exportfs -va** command to do this. It will write the exported directories to **/var/lib/nfs/etab**, where they are immediately applied.

5.8.2.3. client configuration

We have seen the **mount** command and the **/etc/fstab** file before.

```
root@RHELv4u2:~# mount -t nfs barry:/mnt/data/iso /home/project55/
root@RHELv4u2:~# cat /etc/fstab | grep nfs
barry:/mnt/data/iso   /home/iso              nfs     defaults    0 0
root@RHELv4u2:~#
```

5.8.2.4. Mounting NAS

Just a simple fictitious example. Suppose the project55 people tell you they only need a couple of CD-ROM images, and you already have them available on an NFS server. You could issue the following command to mount the network attached storage on their /home/project55 mount point.

```
root@RHELv4u2:~# mount -t nfs 192.168.1.40:/mnt/data/iso /home/project55/
root@RHELv4u2:~# ls -lh /home/project55/
total 3.6G
drwxr-xr-x  2 1000 1000 4.0K Jan 16 17:55 RHELv4u1
drwxr-xr-x  2 1000 1000 4.0K Jan 16 14:14 RHELv4u2
drwxr-xr-x  2 1000 1000 4.0K Jan 16 14:54 RHELv4u3
drwxr-xr-x  2 1000 1000 4.0K Jan 16 11:09 RHELv4u4
-rw-r--r--  1 root root 1.6G Oct 13 15:22 sled10-vmwarews5-vm.zip
root@RHELv4u2:~#
```

5.8.3. Practice NFS

1. Create two directories with some files. Use NFS to share one of them as read only, the other must be writable. Have your neighbour connect to them to test.

2. Investigate the user owner of the files created by your neighbour.

3. Protect a share by ip-address or hostname, so only your neighbour can connect.

Chapter 6. Scheduling

6.1. About scheduling

Linux administrators use the **at** to schedule one time jobs. Recurring jobs are better scheduled with **cron**. The next two sections will discuss both tools.

6.2. at

Simple scheduling can be done with the **at** command. This screenshot shows the scheduling of the date command at 22:01 and the sleep command at 22:03.

```
root@laika:~# at 22:01
at> date
at> <EOT>
job 1 at Wed Aug  1 22:01:00 2007
root@laika:~# at 22:03
at> sleep 10
at> <EOT>
job 2 at Wed Aug  1 22:03:00 2007
root@laika:~#
```

In real life you will hopefully be scheduling more useful commands ;-)

It is easy to check when jobs are scheduled with the **atq** or **at -l** commands.

```
root@laika:~# atq
1       Wed Aug  1 22:01:00 2007 a root
2       Wed Aug  1 22:03:00 2007 a root
root@laika:~# at -l
1       Wed Aug  1 22:01:00 2007 a root
2       Wed Aug  1 22:03:00 2007 a root
root@laika:~#
```

The at command understands English words like tomorrow and teatime to schedule commands the next day and at four in the afternoon.

```
root@laika:~# at 10:05 tomorrow
at> sleep 100
at> <EOT>
job 5 at Thu Aug  2 10:05:00 2007
root@laika:~# at teatime tomorrow
at> tea
at> <EOT>
job 6 at Thu Aug  2 16:00:00 2007
root@laika:~# atq
6       Thu Aug  2 16:00:00 2007 a root
5       Thu Aug  2 10:05:00 2007 a root
root@laika:~#
```

Jobs in the at queue can be removed with **atrm**.

```
root@laika:~# atq
6       Thu Aug  2 16:00:00 2007 a root
5       Thu Aug  2 10:05:00 2007 a root
root@laika:~# atrm 5
root@laika:~# atq
6       Thu Aug  2 16:00:00 2007 a root
root@laika:~#
```

You can also use the **/etc/at.allow** and **/etc/at.deny** files to manage who can schedule jobs with at.

6.3. cron

6.3.1. crontab

The **crontab(1)** command can be used to maintain the **crontab(5)** file. Each user can have their own crontab file to schedule jobs at a specific time. This time can be specified with five fields in this order: minute, hour, day of the month, month and day of the week. If a field contains an asterisk (*), then this means all values of that field.

The following example means : run script42 eight minutes after two, every day of the month, every month and every day of the week.

```
8 14 * * * script42
```

Run script8472 every month on the first of the month at 25 past midnight.

```
25 0 1 * * script8472
```

Run this script33 every two minutes on Sunday (both 0 and 7 refer to Sunday).

```
*/2 * * * 0
```

Instead of these five fields, you can also type one of these: @reboot, @yearly or @annually, @monthly, @weekly, @daily or @midnight, and @hourly.

Users should not edit the crontab file directly, instead they should type **crontab -e** which will use the editor defined in the EDITOR or VISUAL environment variable. Users can display their cron table with **crontab -l**. The **cron daemon** is reading the cron tables, taking into account the **/etc/cron.allow** and **/etc/cron.deny** files.

6.3.2. /etc/cron*

The **/etc/crontab** file contains entries for when to run hourly/daily/weekly/monthly tasks. It will look similar to this output.

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

20 3 * * *        root    run-parts --report /etc/cron.daily
40 3 * * 7        root    run-parts --report /etc/cron.weekly
55 3 1 * *        root    run-parts --report /etc/cron.monthly
```

The directories shown in the next screenshot contain the tasks that are run at the times scheduled in /etc/crontab. The /etc/cron.d directory is for special cases, to schedule jobs that require finer control than hourly/daily/weekly/monthly.

```
paul@laika:~$ ls -ld /etc/cron.*
drwxr-xr-x 2 root root 4096 2008-04-11 09:14 /etc/cron.d
drwxr-xr-x 2 root root 4096 2008-04-19 15:04 /etc/cron.daily
drwxr-xr-x 2 root root 4096 2008-04-11 09:14 /etc/cron.hourly
drwxr-xr-x 2 root root 4096 2008-04-11 09:14 /etc/cron.monthly
drwxr-xr-x 2 root root 4096 2008-04-11 09:14 /etc/cron.weekly
```

6.4. Practice Scheduling

1. Schedule two jobs with at, display the at queue and remove a job.

2. As normal user, use crontab -e to schedule a script to run every four minutes.

3. As root, display the crontab file of your normal user.

4. Take a look at the cron files and directories in /etc and understand them. What is the run-parts command doing ?

Chapter 7. Logging

7.1. About logging

7.1.1. /var/log

The location for log files according to the FHS is **/var/log**. You will find a lot of log files and directories for common applications in /var/log.

```
[paul@RHEL4b ~]$ ls /var/log
acpid           cron.2    maillog.2   quagga           secure.4
amanda          cron.3    maillog.3   radius           spooler
anaconda.log    cron.4    maillog.4   rpmpkgs          spooler.1
anaconda.syslog cups      mailman     rpmpkgs.1        spooler.2
anaconda.xlog   dmesg     messages    rpmpkgs.2        spooler.3
audit           exim      messages.1  rpmpkgs.3        spooler.4
boot.log        gdm       messages.2  rpmpkgs.4        squid
boot.log.1      httpd     messages.3  sa               uucp
boot.log.2      iiim      messages.4  samba            vbox
boot.log.3      iptraf    mysqld.log  scrollkeeper.log vmware-tools-guestd
boot.log.4      lastlog   news        secure           wtmp
canna           mail      pgsql       secure.1         wtmp.1
cron            maillog   ppp         secure.2         Xorg.0.log
cron.1          maillog.1 prelink.log secure.3         Xorg.0.log.old
[paul@RHEL4b ~]$
```

7.1.2. /var/log/messages

A typical first file to check when troubleshooting is the **/var/log/messages** file. By default this file will contain information on what just happened to the system.

```
[root@RHEL4b ~]# tail /var/log/messages
Jul 30 05:13:56 anacron: anacron startup succeeded
Jul 30 05:13:56 atd: atd startup succeeded
Jul 30 05:13:57 messagebus: messagebus startup succeeded
Jul 30 05:13:57 cups-config-daemon: cups-config-daemon startup succeeded
Jul 30 05:13:58 haldaemon: haldaemon startup succeeded
Jul 30 05:14:00 fstab-sync[3560]: removed all generated mount points
Jul 30 05:14:01 fstab-sync[3628]: added mount point /media/cdrom for...
Jul 30 05:14:01 fstab-sync[3646]: added mount point /media/floppy for...
Jul 30 05:16:46 sshd(pam_unix)[3662]: session opened for user paul by...
Jul 30 06:06:37 su(pam_unix)[3904]: session opened for user root by paul
[root@RHEL4b ~]#
```

7.2. Login logging

To keep track of who is logging into the system, Linux can maintain the **/var/log/wtmp**, **/var/log/btmp**, **/var/run/utmp** and **/var/log/lastlog** files.

7.2.1. /var/run/utmp (who)

Use the **who** command to see the /var/run/utmp file. This command is showing you all the **currently** logged in users. Notice that the utmp file is in /var/run and not in /var/log .

```
[root@rhel4 ~]# who
paul     pts/1        Feb 14 18:21 (192.168.1.45)
sandra   pts/2        Feb 14 18:11 (192.168.1.42)
inge     pts/3        Feb 14 12:01 (192.168.1.33)
els      pts/4        Feb 14 14:33 (192.168.1.19)
```

7.2.2. /var/log/wtmp (last)

The /var/log/wtmp file is updated by the **login program**. Use **last** to see the /var/run/wtmp file.

```
[root@rhel4a ~]# last | head
paul     pts/1       192.168.1.45     Wed Feb 14 18:39   still logged in
reboot   system boot 2.6.9-42.0.8.ELs Wed Feb 14 18:21          (01:15)
nicolas  pts/5       pc-dss.telematic Wed Feb 14 12:32 - 13:06  (00:33)
stefaan  pts/3       pc-sde.telematic Wed Feb 14 12:28 - 12:40  (00:12)
nicolas  pts/3       pc-nae.telematic Wed Feb 14 11:36 - 12:21  (00:45)
nicolas  pts/3       pc-nae.telematic Wed Feb 14 11:34 - 11:36  (00:01)
dirk     pts/5       pc-dss.telematic Wed Feb 14 10:03 - 12:31  (02:28)
nicolas  pts/3       pc-nae.telematic Wed Feb 14 09:45 - 11:34  (01:48)
dimitri  pts/5       rhel4            Wed Feb 14 07:57 - 08:38  (00:40)
stefaan  pts/4       pc-sde.telematic Wed Feb 14 07:16 - down   (05:50)
[root@rhel4a ~]#
```

The last command can also be used to get a list of last reboots.

```
[paul@rekkie ~]$ last reboot
reboot   system boot  2.6.16-rekkie   Mon Jul 30 05:13     (370+08:42)

wtmp begins Tue May 30 23:11:45 2006
[paul@rekkie ~]$
```

7.2.3. /var/log/lastlog (lastlog)

Use **lastlog** to see the /var/log/lastlog file.

```
[root@rhel4a ~]# lastlog | tail
tim              pts/5  10.170.1.122     Tue Feb 13 09:36:54 +0100 2007
rm               pts/6  rhel4            Tue Feb 13 10:06:56 +0100 2007
henk                                     **Never logged in**
stefaan          pts/3  pc-sde.telematic Wed Feb 14 12:28:38 +0100 2007
dirk             pts/5  pc-dss.telematic Wed Feb 14 10:03:11 +0100 2007
```

```
arsene                                  **Never logged in**
nicolas          pts/5  pc-dss.telematic Wed Feb 14 12:32:18 +0100 2007
dimitri          pts/5  rhel4            Wed Feb 14 07:57:19 +0100 2007
bashuserrm       pts/7  rhel4            Tue Feb 13 10:35:40 +0100 2007
kornuserrm       pts/5  rhel4            Tue Feb 13 10:06:17 +0100 2007
[root@rhel4a ~]#
```

7.2.4. /var/log/btmp (lastb)

There is also the **lastb** command to display the **/var/log/btmp** file. This file is updated by the login program when entering the wrong password, so it contains failed login attempts. Many computers will not have this file, resulting in no logging of failed login attempts.

```
[root@RHEL4b ~]# lastb
lastb: /var/log/btmp: No such file or directory
Perhaps this file was removed by the operator to prevent logging lastb\
 info.
[root@RHEL4b ~]#
```

The reason given for this is that users sometimes type their password by mistake instead of their login, so this world readable file poses a security risk. You can enable bad login logging by simply creating the file. Doing a chmod o-r /var/log/btmp improves security.

```
[root@RHEL4b ~]# touch /var/log/btmp
[root@RHEL4b ~]# ll /var/log/btmp
-rw-r--r--  1 root root 0 Jul 30 06:12 /var/log/btmp
[root@RHEL4b ~]# chmod o-r /var/log/btmp
[root@RHEL4b ~]# lastb

btmp begins Mon Jul 30 06:12:19 2007
[root@RHEL4b ~]#
```

Failed logins via ssh, rlogin or su are not registered in /var/log/btmp. Failed logins via tty are.

```
[root@RHEL4b ~]# lastb
HalvarFl tty3                  Mon Jul 30 07:10 - 07:10  (00:00)
Maria    tty1                  Mon Jul 30 07:09 - 07:09  (00:00)
Roberto  tty1                  Mon Jul 30 07:09 - 07:09  (00:00)

btmp begins Mon Jul 30 07:09:32 2007
[root@RHEL4b ~]#
```

7.2.5. su and ssh logins

Depending on the distribution, you may also have the **/var/log/secure** file being filled with messages from the auth and/or authpriv syslog facilities. This log will include

su and/or ssh failed login attempts. Some distributions put this in **/var/log/auth.log**, verify the syslog configuration.

```
[root@RHEL4b ~]# cat /var/log/secure
Jul 30 07:09:03 sshd[4387]: Accepted publickey for paul from ::ffff:19\
2.168.1.52 port 33188 ssh2
Jul 30 05:09:03 sshd[4388]: Accepted publickey for paul from ::ffff:19\
2.168.1.52 port 33188 ssh2
Jul 30 07:22:27 sshd[4655]: Failed password for Hermione from ::ffff:1\
92.168.1.52 port 38752 ssh2
Jul 30 05:22:27 sshd[4656]: Failed password for Hermione from ::ffff:1\
92.168.1.52 port 38752 ssh2
Jul 30 07:22:30 sshd[4655]: Failed password for Hermione from ::ffff:1\
92.168.1.52 port 38752 ssh2
Jul 30 05:22:30 sshd[4656]: Failed password for Hermione from ::ffff:1\
92.168.1.52 port 38752 ssh2
Jul 30 07:22:33 sshd[4655]: Failed password for Hermione from ::ffff:1\
92.168.1.52 port 38752 ssh2
Jul 30 05:22:33 sshd[4656]: Failed password for Hermione from ::ffff:1\
92.168.1.52 port 38752 ssh2
Jul 30 08:27:33 sshd[5018]: Invalid user roberto from ::ffff:192.168.1\
.52
Jul 30 06:27:33 sshd[5019]: input_userauth_request: invalid user rober\
to
Jul 30 06:27:33 sshd[5019]: Failed none for invalid user roberto from \
::ffff:192.168.1.52 port 41064 ssh2
Jul 30 06:27:33 sshd[5019]: Failed publickey for invalid user roberto \
from ::ffff:192.168.1.52 port 41064 ssh2
Jul 30 08:27:36 sshd[5018]: Failed password for invalid user roberto f\
rom ::ffff:192.168.1.52 port 41064 ssh2
Jul 30 06:27:36 sshd[5019]: Failed password for invalid user roberto f\
rom ::ffff:192.168.1.52 port 41064 ssh2
[root@RHEL4b ~]#
```

You can enable this yourself, with a custom logfile by adding the following line tot syslog.conf.

```
auth.*,authpriv.*                  /var/log/customsec.log
```

7.3. Syslogd daemon

7.3.1. About syslog

The standard method of logging on Linux is through the **syslogd** daemon. Syslog was developed by **Eric Allman** for sendmail, but quickly became a standard among many Unix applications and was much later written as rfc 3164. The syslog daemon can receive messages on udp **port 514** from many applications (and appliances), and can append to logfiles, print, display messages on terminals and forward logs to other syslogd daemons on other machines. The syslogd daemon is configured in **/etc/syslog.conf**.

Each line in the configuration file uses a **facility** to determine where the message is coming from. It also contains a **level** for the severity of the message, and an **action** to decide on what to do with the message.

7.3.2. Facilities

The **man syslog.conf** will explain the different default facilities for certain daemons, such as mail, lpr, news and kern(el) messages. The local0 to local7 facility can be used for appliances (or any networked device that supports syslog). Here is a list of all facilities for syslog.conf version 1.3. The security keyword is deprecated.

```
auth (security)
authpriv
cron
daemon
ftp
kern
lpr mail
mark (internal use only)
news
syslog
user
uucp
local0-7
```

7.3.3. Levels

The worst severity a message can have is **emerg** followed by **alert** and **crit**. Lowest priority should go to **info** and **debug** messages. Specifying a severity will also log all messages with a higher severity. You can prefix the severity with = to obtain only messages that match that severity. You can also specify **.none** to prevent a specific action from any message from a certain facility.

Here is a list of all levels, in ascending order. The keywords warn, error and panic are deprecated.

```
debug
info
notice
warning (warn)
err (error)
crit
alert
emerg (panic)
```

7.3.4. Actions

The default action is to send a message to the username listed as action. When the action is prefixed with a / then syslog will send the message to the file (which can be a regular file, but also a printer or terminal). The **@** sign prefix will send the message on to another syslog server. Here is a list of all possible actions.

```
root,user1      list of users, seperated by comma's
*               message to all logged on users
/               file (can be a printer, a console, a tty, ...)
-/              file, but don't sync after every write
|               named pipe
@               other syslog hostname
```

In addition, you can prefix actions with a - to omit syncing the file after every logging.

7.3.5. Configuration

Below a sample configuration of custom local4 messages in **/etc/syslog.conf**.

```
local4.crit                             /var/log/critandabove
local4.=crit                            /var/log/onlycrit
local4.*                                /var/log/alllocal4
```

Don't forget to restart the server.

```
[root@rhel4a ~]# /etc/init.d/syslog restart
Shutting down kernel logger:                               [  OK  ]
Shutting down system logger:                               [  OK  ]
Starting system logger:                                    [  OK  ]
Starting kernel logger:                                    [  OK  ]
[root@rhel4a ~]#
```

7.4. logger

The logger command can be used to generate syslog test messages. You can aslo use it in scripts. An example of testing syslogd with the **logger** tool.

```
[root@rhel4a ~]# logger -p local4.debug "l4 debug"
[root@rhel4a ~]# logger -p local4.crit "l4 crit"
[root@rhel4a ~]# logger -p local4.emerg "l4 emerg"
[root@rhel4a ~]#
```

The results of the tests with logger.

```
[root@rhel4a ~]# cat /var/log/critandabove
Feb 14 19:55:19 rhel4a paul: l4 crit
Feb 14 19:55:28 rhel4a paul: l4 emerg
[root@rhel4a ~]# cat /var/log/onlycrit
Feb 14 19:55:19 rhel4a paul: l4 crit
[root@rhel4a ~]# cat /var/log/alllocal4
Feb 14 19:55:11 rhel4a paul: l4 debug
Feb 14 19:55:19 rhel4a paul: l4 crit
Feb 14 19:55:28 rhel4a paul: l4 emerg
[root@rhel4a ~]#
```

7.5. Watching logs

You might want to use the **tail -f** command to look at the last lines of a log file. The -f option will dynamically display lines that are appended to the log. You can do the same with other commands by preceding them with the **watch** command.

7.6. Rotating logs

A lot of log files are always growing in size. To keep this within bounds, you might want to use **logrotate** to rotate, compress, remove and mail logfiles. More info on the logrotate command in the scheduling chapter.

7.7. Practice Logging

1. Display the /var/run/utmp file.

2. Display the /var/log/wtmp file.

3. Use the lastlog and lastb commands, understand the difference.

4. Examine syslog to find the location of the log file containing ssh failed logins.

5. Configure syslog to put local4.error and above messages in /var/log/l4e.log and local4.info only .info in /var/log/l4i.log. Test that it works with the logger tool!

6. Configure /var/log/Mysu.log, all the su to root messages should go in that log. Test that it works!

7. Send the local5 messages to the syslog server of your neighbour. Test that it works.

8. Write a script that executes logger to local4 every 5 seconds (different message). Use tail -f and watch on your local4 log files.

Chapter 8. Memory Management

8.1. About Memory

You can display information about RAM memory with **free -om**, **top** and cat **/proc/meminfo**. You should understand terms like **swapping**, **paging** and **virtual memory**.

8.2. Swap space

8.2.1. About swap space

When the operating system needs more memory than physically present in RAM, it will use **swap space**. Swap space is located on slower but cheaper memory. Notice that, although hard disks are commonly used for swap space, their access times are one hundred thousand times slower.

The swap space can be a file, a partition, or a combination of files and partitions. You can see the swap space with the **free** command, or with **cat /proc/swaps**.

```
paul@RHELv4u4:~$ free -om
             total       used       free     shared    buffers     cached
Mem:            249        245          4          0        125         55
Swap:          1023          0       1023
paul@RHELv4u4:~$ cat /proc/swaps
Filename                                Type            Size    Used    Priority
/dev/mapper/VolGroup00-LogVol01         partition       1048568 0       -1
paul@RHELv4u4:~$
```

The amount of swap space that you need depends heavily on the services that the computer provides.

8.2.2. Creating a swap partition

You can activate or deactivate swap space with the **swapon** and **swapoff** commands. New swap space can be created with the **mkswap** command. The screenshot below shows the creation and activation of a swap partition.

```
root@RHELv4u4:~# fdisk -l 2> /dev/null | grep hda
Disk /dev/hda: 536 MB, 536870912 bytes
/dev/hda1               1        1040      524128+  83  Linux
root@RHELv4u4:~# mkswap /dev/hda1
Setting up swapspace version 1, size = 536702 kB
root@RHELv4u4:~# swapon /dev/hda1
```

Now you can see that **/proc/swaps** displays all swap spaces separately, whereas the **free -om** command only makes a human readable summary.

```
root@RHELv4u4:~# cat /proc/swaps
Filename                        Type         Size    Used    Priority
/dev/mapper/VolGroup00-LogVol01 partition    1048568 0       -1
/dev/hda1                       partition    524120  0       -2
root@RHELv4u4:~# free -om
             total       used       free     shared    buffers     cached
Mem:           249        245          4          0        125         54
Swap:         1535          0       1535
root@RHELv4u4:~#
```

8.2.3. Creating a swap file

Here is one more example showing you how to create a **swap file**. On Solaris you can use **mkfile** instead of **dd**.

```
root@RHELv4u4:~# dd if=/dev/zero of=/smallswapfile bs=1024 count=4096
4096+0 records in
4096+0 records out
root@RHELv4u4:~# mkswap /smallswapfile
Setting up swapspace version 1, size = 4190 kB
root@RHELv4u4:~# swapon /smallswapfile
root@RHELv4u4:~# cat /proc/swaps
Filename                        Type         Size    Used    Priority
/dev/mapper/VolGroup00-LogVol01 partition    1048568 0       -1
/dev/hda1                       partition    524120  0       -2
/smallswapfile                  file         4088    0       -3
root@RHELv4u4:~#
```

8.2.4. Swap space in /etc/fstab

If you like these swaps to be permanent, then don't forget to add them to **/etc/fstab**. The lines in /etc/fstab will be similar to the following.

```
/dev/hda1         swap       swap     defaults      0 0
/smallswapfile    swap       swap     defaults      0 0
```

8.3. Practice Memory

1. Use **dmesg** to find the total amount of memory in your computer.

2. Use **free** to display memory usage in kilobytes (then in megabytes).

3. On the Red Hat, create a swap partition on one of your new disks, and a swap file on the other new disk.

4. Put all swap spaces in /etc/fstab and activate them. Use free again to verify that it works.

Chapter 9. Installing Linux

9.1. About

The past couple of years the installation of linux has become a lot easier then before, at least for end users installing a distro like Ubuntu, Fedora, Debian or Mandrake on their home computer. Servers usually come pre-installed, and if not pre-installed, then setup of a linux server today is very easy.

Linux can be installaed in many different ways. End users most commonly use cdrom's or dvd's for installation, most of the time with a working internet connection te receive updates. Administrators might prefer network installations using protocols like **tftp**, **bootp**, **rarp** and/or **nfs** or response file solutions like **Red Hat Kickstart** or **Solaris Jumpstart**.

9.2. Installation by cdrom

Installation of linux from cdrom is easy! Most distributions ask very few questions during install (keyboard type, language, username) and detect all the hardware themselves. There is usually no need to retrieve thrid-party drivers from the internet. The GUI installation gives options like Desktop (for end users), Workstation (for developers), Server or minimal (usually without graphical interface).

9.3. Installation with rarp and tftp

Installing over the network involves powering on the machine, have it find a rarpd server to get an ip-address, then let it find an tftps server to get an installation image copied to the machine. This image can then boot. The procedure below demonstrates how to setup three Sun SPARC servers with Ubuntu Linux, using a Debian Linux machine to host the tftp, bootp and nfs daemons.

First we need to configure the mac to ip resolution in the **/etc/ethers** configuration file. Each server will receive a unique ip-address during installation.

```
root@laika:~# cat /etc/ethers
00:03:ba:02:c3:82       192.168.1.71
00:03:ba:09:7c:f9       192.168.1.72
00:03:ba:09:7f:d2       192.168.1.73
```

We need to install the rarpd and tftpd daemons on the (Debian) machine that will be hosting the install image.

```
root@laika:~# aptitude install rarpd
root@laika:~# aptitude install tftpd
```

The tftp services must be activated in inetd or xinetd.

```
root@laika:~# cat /etc/inetd.conf | tail -1
tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd /srv/tftp
```

And finally the linux install image must be present in the tftp served directory. The filename of the image must be the hex ip-address, this is accomplished with symbolic links.

```
root@laika:~# ll /srv/tftp/
total 7.5M
lrwxrwxrwx 1 root root   13 2007-03-02 21:49 C0A80147 -> ubuntu610.img
lrwxrwxrwx 1 root root   13 2007-03-03 14:13 C0A80148 -> ubuntu610.img
lrwxrwxrwx 1 root root   13 2007-03-02 21:49 C0A80149 -> ubuntu610.img
-rw-r--r-- 1 paul paul 7.5M 2007-03-02 21:42 ubuntu610.img
```

Time to enter **boot net** now in the openboot prompt. Twenty minutes later the three servers where humming with linux.

9.4. About Red Hat Kickstart

Automating Linux installations with response files can be done with **Red Hat kickstart**. One way to set it up is by using the graphical tool **/usr/sbin/system-config-kickstart**. If you prefer to set it up manually, read on.

You can modify the sample kickstart file **RH-DOCS/sample.ks** (can be found on the documentation dvd). Put this file so **anaconda** can read it.

Anaconda is the Red Ha installer written in Python. The name is chose because anacondas are lizard-eating pythons. Lizard is the name of the Caldera Linux installation program.

Another option is to start with the **/root/anaconda-ks.cfg** file. This is a sample kickstart file that contains all the settings from your current installation.

Do not change the order of the sections inside your kickstart file! The Red Hat System Administration Guide contains about 25 pages describing all the options, most of them are easy ti understand if you already performed a couple of installations.

9.5. Using Kickstart

To use kickstart, name your kickstart file **ks.cfg** and put it in the root directory of your installation cdrom (or on a usb stick or a floppy). For network based installations, name the file **$ip-address-kickstart** and place the following in **dhcpd.conf**.

```
filename "/export/kickstart"
next-server remote.installation.server
```

Leaving out the **next-server** line will result in the client looking for the file on the dhcp server itself.

Booting from cdrom with kickstart reauires the following commnad at the **boot:** prompt.

```
linux ks=cdrom:/ks.cfg
```

When the kickstart file is on the network, use nfs or http like in these examples.

```
linux ks=nfs:servername:/path/to/ks.cfg
```

```
linux ks=http://servername/path/to/ks.cfg
```

Chapter 10. Package Management

10.1. About repositories

Software for your Linux distribution is not scattered all over the place, it is in general managed in a central distributed repository. This means that applications in the repository are tested for your distribution. Installing this software is very easy. The problem begins when you need software from outside of the central repository.

You can install software from the repository on Linux in different ways. Beginners should use the graphical software installation tool that is provided by the distribution (Synaptic on Debian, Add/Remove Software on Ubuntu, Yast on Suse, ...). More advanced people tend to use the command line (rpm, yum, dpkg, aptitude). A third option is to download vanilla source code and compile the software yourself, providing the application is open source.

10.2. Red Hat Package Manager

The **Red Hat Package Manager** is used by many distributions. Besides Red Hat, who originally created the **.rpm** format and Fedora, there is also Mandriva, Red Flag Linux, OpenSUSE and others. The most common front ends (besides **rpm**) using this tool are **yum**, **YaST** and **up2date**.

10.2.1. rpm

10.2.1.1. About rpm

The **Red Hat Package Manager** can be used on the command line with **rpm** or in a graphical way going to Applications--System Settings--Add/Remove Applications. Type **rpm --help** to see some of the options.

Software distributed in the rpm format will be named **foo-version.platform.rpm** .

10.2.1.2. rpm -qa

To obtain a list of all installed software, use the **rpm -qa** command.

```
[root@RHEL52 ~]# rpm -qa | grep samba
system-config-samba-1.2.39-1.el5
samba-3.0.28-1.el5_2.1
samba-client-3.0.28-1.el5_2.1
samba-common-3.0.28-1.el5_2.1
```

10.2.1.3. rpm -q

To verify whether one package is installed, use **rpm -q**.

```
root@RHELv4u4:~# rpm -q gcc
gcc-3.4.6-3
root@RHELv4u4:~# rpm -q laika
package laika is not installed
```

10.2.1.4. rpm -q --redhatprovides

To check whether a package is provided by Red Hat, use the **--redhatprovides** option.

```
root@RHELv4u4:~# rpm -q --redhatprovides bash
bash-3.0-19.3
root@RHELv4u4:~# rpm -q --redhatprovides gcc
gcc-3.4.6-3
root@RHELv4u4:~# rpm -q --redhatprovides laika
no package provides laika
```

10.2.1.5. rpm -Uvh

To install or upgrade a package, use the -Uvh switches. The -U switch is the same as -i for install, except that older versions of the software are removed. The -vh switches are for nicer output.

```
root@RHELv4u4:~# rpm -Uvh gcc-3.4.6-3
```

10.2.1.6. rpm -e

To remove a package, use the -e switch.

```
root@RHELv4u4:~# rpm -e gcc-3.4.6-3
```

rpm -e verifies dependencies, and thus will prevent you from accidentaly erasing packages that are needed by other packages.

```
[root@RHEL52 ~]# rpm -e gcc-4.1.2-42.el5
error: Failed dependencies:
gcc = 4.1.2-42.el5 is needed by (installed) gcc-c++-4.1.2-42.el5.i386
gcc = 4.1.2-42.el5 is needed by (installed) gcc-gfortran-4.1.2-42.el5.i386
gcc is needed by (installed) systemtap-0.6.2-1.el5_2.2.i386
```

10.2.1.7. /var/lib/rpm

The **rpm** database is located at **/var/lib/rpm**. This database contains all meta information about packages that are installed (via rpm). It keeps track of all files, which enables complete removes of software.

10.2.2. yum

The **Yellowdog Updater, Modified (yum)** is an easier command to work with rpm packages. It is installed by default on Fedora and is optional on Red Hat Enterprise Linux.

Issue **yum list available** to see a list of available packages.

```
yum list available
```

To search for a package containing a certain string in the description or name use **yum search $string**.

```
yum search $string
```

To install an application, use **yum install $package**. Naturally yum will install all the necesary dependencies.

```
yum install $package
```

To bring all applications up to date, by downloading and installing them, issue **yum update**. All software that was installed via yum will be updated to the latest version that is available in the repository.

```
yum update
```

The configuration of your yum repositories is done in **/etc/yum/yum.conf** and **/etc/yum/repos.d/**.

10.2.3. up2date

up2date is the Red Hat Update Agent. It is available on Red Hat Enterprise Linux and serves as a connection to RHN (Red Hat Network). It has simple switches like -d for download, -i for install and -l for list (of packages that can be updated).

10.3. Debian Package Management

10.3.1. About .deb

The Debian Package Management System is used by Debian, Ubuntu, Linux Mint and all derivatives from Debian and Ubuntu. It uses **.deb** packages.

10.3.2. dpkg -l

The low level tool to work with **.deb** packages is **dpkg**. Here you see how to obtain a list of all installed packages. The ii at the beginning means the package is installed.

```
root@laika:~# dpkg -l | grep gcc-4.2
ii gcc-4.2      4.2.4-1ubuntu3 The GNU C compiler
ii gcc-4.2-base 4.2.4-1ubuntu3 The GNU Compiler Collection (base package)
```

10.3.3. dpkg

You could use **dpkg -i** to install a package and **dpkg -r** to remove a package, but you'd have to manually keep track of dependencies.

10.3.4. aptitude

Most people use **aptitude** for package management on Debian and Ubuntu Systems.

To synchronize with the repositories.

```
aptitude update
```

To patch and upgrade all software to the latest version..

```
aptitude safe-upgrade
```

To install an application with all dependencies.

```
aptitude install $package
```

To search the repositories for applications that contain a certain string in their name or description.

```
aptitude search $string
```

To remove an application and all unused files.

```
aptitude remove $package
```

10.4. Downloading software

First and most important, whenever you download software, start by reading the README file!

Normally the readme will explain what to do after download. You will probably receive a .tar.gz or a .tgz file. Read the documentation, then put the compressed file in a directory. You can use the following to find out where the package wants to install.

```
tar tvzpf $downloadedFile.tgz
```

You unpack them like with **tar xzf**, it will create a directory called applicationName-1.2.3

```
tar xzf $applicationName.tgz
```

Replace the z with a j when the file ends in .tar.bz2. The **tar**, gzip and bzip2 commands are explained in detail later.

If you download a .deb file, then you'll have to use dpkg to install it, .rpm's can be installed with the rpm command. Sometimes people use the **alien** command to convert between package formats.

10.5. Compiling software

First and most important, whenever you download source code for installation, start by reading the README file!

Usually the steps are always the same three : running **./configure** followed by **make** (which is the actual compiling) and then by **make install** to copy the files to their proper location.

```
./configure
make
make install
```

10.6. Practice: Installing software

1. Find the Graphical application on all computers to add and remove applications.

2. Verify on both systems whether gcc is installed.

3. Use aptitude or yum to search for and install the 'dict', 'samba' and 'wesnoth' applications. Did you find all them all ?

4. Search the internet for 'webmin' and install it.

5. If time permits, uninstall Samba from the ubuntu machine, download the latest version from samba.org and install it.

10.7. Solution: Installing software

1. Find the Graphical application on all computers to add and remove applications.

2. Verify on both systems whether gcc is installed.

```
dpkg -l | grep gcc
```

```
rpm -qa | grep gcc
```

3. Use aptitude or yum to search for and install the 'dict', 'samba' and 'wesnoth' applications. Did you find all them all ?

```
aptitude search wesnoth (Debian, Ubuntu and family)
```

```
yum search wesnoth (Red Hat and family)
```

4. Search the internet for 'webmin' and install it.

```
Google should point you to webmin.com.
```

```
There are several formats available there choose .rpm, .deb or .tgz .
```

5. If time permits, uninstall Samba from the ubuntu machine, download the latest version from samba.org and install it.

Chapter 11. Backup

11.1. About tape devices

Don't forget that the name of a device strictly speaking has no meaning since the kernel will use the major and minor number to find the hardware! See the man page of **mknod** and the devices.txt file in the linux kernel source for more info.

11.1.1. SCSI tapes

On the official Linux device list (http://www.lanana.org/docs/device-list/) we find the names for SCSI tapes (major 9 char). SCSI tape devices are located underneath **/dev/st** and are numbered starting with 0 for the first tape device.

```
/dev/st0   First tape device
/dev/st1   Second tape device
/dev/st2   Third tape device
```

To prevent **automatic rewinding of tapes**, prefix them with the letter n.

```
/dev/nst0   First no rewind tape device
/dev/nst1   Second no rewind tape device
/dev/nst2   Third no rewind tape device
```

By default, SCSI tapes on linux will use the highest hardware compression that is supported by the tape device. To lower the compression level, append one of the letters l (low), m (medium) or a (auto) to the tape name.

```
/dev/st0l   First low compression tape device
/dev/st0m   First medium compression tape device
/dev/nst2m  Third no rewind medium compression tape device
```

11.1.2. IDE tapes

On the official Linux device list (http://www.lanana.org/docs/device-list/) we find the names for IDE tapes (major 37 char). IDE tape devices are located underneath **/dev/ht** and are numbered starting with 0 for the first tape device. No rewind and compression is similar to SCSI tapes.

```
/dev/ht0   First IDE tape device
/dev/nht0  Second no rewind IDE tape device
/dev/ht0m  First medium compression IDE tape device
```

11.1.3. mt

To manage your tapes, use **mt** (Magnetic Tape). Some examples.

To receive information about the status of the tape.

```
mt -f /dev/st0 status
```

To rewind a tape...

```
mt -f /dev/st0 rewind
```

To rewind and eject a tape...

```
mt -f /dev/st0 eject
```

To erase a tape...

```
mt -f /dev/st0 erase
```

11.2. Compression

It can be beneficial to compress files before backup. The two most popular tools for compression of regular files on linux are **gzip/gunzip** and **bzip2/bunzip2**. Below you can see gzip in action, notice that it adds the **.gz** extension to the file.

```
paul@RHELv4u4:~/test$ ls -l allfiles.tx*
-rw-rw-r--  1 paul paul 8813553 Feb 27 05:38 allfiles.txt
paul@RHELv4u4:~/test$ gzip allfiles.txt
paul@RHELv4u4:~/test$ ls -l allfiles.tx*
-rw-rw-r--  1 paul paul 931863 Feb 27 05:38 allfiles.txt.gz
paul@RHELv4u4:~/test$ gunzip allfiles.txt.gz
paul@RHELv4u4:~/test$ ls -l allfiles.tx*
-rw-rw-r--  1 paul paul 8813553 Feb 27 05:38 allfiles.txt
paul@RHELv4u4:~/test$
```

In general, gzip is much faster than bzip2, but the latter one compresses a lot better. Let us compare the two.

```
paul@RHELv4u4:~/test$ cp allfiles.txt bllfiles.txt
paul@RHELv4u4:~/test$ time gzip allfiles.txt

real    0m0.050s
user    0m0.041s
sys     0m0.009s
paul@RHELv4u4:~/test$ time bzip2 bllfiles.txt

real    0m5.968s
user    0m5.794s
sys     0m0.076s
paul@RHELv4u4:~/test$ ls -l ?llfiles.tx*
-rw-rw-r--  1 paul paul 931863 Feb 27 05:38 allfiles.txt.gz
-rw-rw-r--  1 paul paul 708871 May 12 10:52 bllfiles.txt.bz2
paul@RHELv4u4:~/test$
```

11.3. tar

The **tar** utility gets its name from **Tape ARchive**. This tool will receive and send files to a destination (typically a tape or a regular file). The c option is used to create a tar archive (or tarfile), the f option to name/create the **tarfile**. The example below takes a backup of /etc into the file /backup/etc.tar .

```
root@RHELv4u4:~# tar cf /backup/etc.tar /etc
root@RHELv4u4:~# ls -l /backup/etc.tar
-rw-r--r--  1 root root 47800320 May 12 11:47 /backup/etc.tar
root@RHELv4u4:~#
```

Compression can be achieved without pipes since tar uses the z flag to compress with gzip, and the j flag to compress with bzip2.

```
root@RHELv4u4:~# tar czf /backup/etc.tar.gz /etc
root@RHELv4u4:~# tar cjf /backup/etc.tar.bz2 /etc
root@RHELv4u4:~# ls -l /backup/etc.ta*
-rw-r--r--  1 root root 47800320 May 12 11:47 /backup/etc.tar
-rw-r--r--  1 root root  6077340 May 12 11:48 /backup/etc.tar.bz2
-rw-r--r--  1 root root  8496607 May 12 11:47 /backup/etc.tar.gz
root@RHELv4u4:~#
```

The t option is used to **list the contents of a tar file**. Verbose mode is enabled with v (also useful when you want to see the files being archived during archiving).

```
root@RHELv4u4:~# tar tvf /backup/etc.tar
drwxr-xr-x root/root         0 2007-05-12 09:38:21 etc/
-rw-r--r-- root/root      2657 2004-09-27 10:15:03 etc/warnquota.conf
-rw-r--r-- root/root     13136 2006-11-03 17:34:50 etc/mime.types
drwxr-xr-x root/root         0 2004-11-03 13:35:50 etc/sound/
...
```

To **list a specific file in a tar archive**, use the t option, added with the filename (without leading /).

```
root@RHELv4u4:~# tar tvf /backup/etc.tar etc/resolv.conf
-rw-r--r-- root/root        77 2007-05-12 08:31:32 etc/resolv.conf
root@RHELv4u4:~#
```

Use the x flag to **restore a tar archive**, or a single file from the archive. Remember that by default tar will restore the file in the current directory.

```
root@RHELv4u4:~# tar xvf /backup/etc.tar etc/resolv.conf
etc/resolv.conf
root@RHELv4u4:~# ls -l /etc/resolv.conf
-rw-r--r--  2 root root 40 May 12 12:05 /etc/resolv.conf
```

```
root@RHELv4u4:~# ls -l etc/resolv.conf
-rw-r--r--  1 root root 77 May 12 08:31 etc/resolv.conf
root@RHELv4u4:~#
```

You can **preserve file permissions** with the p flag. And you can exclude directories or file with **--exclude**.

```
root ~# tar cpzf /backup/etc_with_perms.tgz /etc
root ~# tar cpzf /backup/etc_no_sysconf.tgz /etc --exclude /etc/sysconfig
root ~# ls -l /backup/etc_*
-rw-r--r--  1 root root 8434293 May 12 12:48 /backup/etc_no_sysconf.tgz
-rw-r--r--  1 root root 8496591 May 12 12:48 /backup/etc_with_perms.tgz
root ~#
```

You can also create a text file with names of files and directories to archive, and then supply this file to tar with the -T flag.

```
root@RHELv4u4:~# find /etc -name *.conf > files_to_archive.txt
root@RHELv4u4:~# echo /home -iname *.pdf >> files_to_archive.txt
root@RHELv4u4:~# tar cpzf /backup/backup.tgz -T files_to_archive.txt
```

The tar utility can receive filenames from the find command, with the help of xargs.

```
find /etc -type f -name "*.conf" | xargs tar czf /backup/confs.tar.gz
```

You can also use tar to copy a directory, this is more efficient than using cp -r.

```
(cd /etc; tar -cf - . ) | (cd /backup/copy_of_etc/; tar -xpf - )
```

Another example of tar, this copies a directory securely over the network.

```
(cd /etc;tar -cf - . )|(ssh user@srv 'cd /backup/cp_of_etc/; tar -xf - ')
```

tar can be used together with gzip and copy a file to a remote server through ssh

```
cat backup.tar | gzip | ssh bashuser@192.168.1.105 "cat - > backup.tgz"
```

Compress the tar backup when it is on the network, but leave it uncompressed at the destination.

```
cat backup.tar | gzip | ssh user@192.168.1.105 "gunzip|cat - > backup.tar"
```

Same as the previous, but let ssh handle the compression

```
cat backup.tar | ssh -C bashuser@192.168.1.105 "cat - > backup.tar"
```

11.4. Backup Types

Linux uses **multilevel incremental** backups using distinct levels. A full backup is a backup at level 0. A higher level x backup will include all changes since the last level x-1 backup.

Suppose you take a full backup on Monday (level 0) and a level 1 backup on Tuesday, then the Tuesday backup will contain all changes since Monday. Taking a level 2 on Wednesday will contain all changes since Tuesday (the last level 2-1). A level 3 backup on Thursday will contain all changes since Wednesday (the last level 3-1). Another level 3 on Friday will also contain all changes since Wednesday. A level 2 backup on Saturday would take all changes since the last level 1 from Tuesday.

11.5. dump and restore

While **dump** is similar to tar, it is also very different because it looks at the file system. Where tar receives a lists of files to backup, dump will find files to backup by itself by examining ext2. Files found by dump will be copied to a tape or regular file. In case the target is not big enough to hold the dump (end-of-media), it is broken into multiple volumes.

Restoring files that were backed up with dump is done with the **restore** command. In the example below we take a full level 0 backup of two partitions to a SCSI tape. The no rewind is mandatory to put the volumes behind each other on the tape.

```
dump 0f /dev/nst0 /boot
dump 0f /dev/nst0 /
```

Listing files in a dump archive is done with **dump -t**, and you can compare files with **dump -C**.

You can omit files from a dump by changing the dump attribute with the **chattr** command. The d attribute on ext will tell dump to skip the file, even during a full backup. In the following example, /etc/hosts is excluded from dump archives.

```
chattr +d /etc/hosts
```

To restore the complete file system with **restore**, use the -r option. This can be useful to change the size or block size of a file system. You should have a clean file system mounted and cd'd into it. Like this example shows.

```
mke2fs /dev/hda3
mount /dev/hda3 /mnt/data
cd /mnt/data
```

```
restore rf /dev/nst0
```

To extract only one file or directory from a dump, use the -x option.

```
restore -xf /dev/st0 /etc
```

11.6. cpio

Different from tar and dump is **cpio** (Copy Input and Output). It can be used to receive filenames, but copies the actual files. This makes it an easy companion with find! Some examples below.

find sends filenames to cpio, which puts the files in an archive.

```
find /etc -depth -print | cpio -oaV -O archive.cpio
```

The same, but compressed with gzip

```
find /etc -depth -print | cpio -oaV | gzip -c > archive.cpio.gz
```

Now pipe it through ssh (backup files to a compressed file on another machine)

```
find /etc -depth -print|cpio -oaV|gzip -c|ssh server "cat - > etc.cpio.gz"
```

find sends filenames to cpio | cpio sends files to ssh | ssh sends files to cpio 'cpio extracts files'

```
find /etc -depth -print | cpio -oaV | ssh user@host 'cpio -imVd'
```

the same but reversed: copy a dir from the remote host to the local machine

```
ssh user@host "find path -depth -print | cpio -oaV" | cpio -imVd
```

11.7. dd

11.7.1. About dd

Some people use **dd** to create backups. This can be very powerful, but dd backups can only be restored to very similar partitions or devices. There are however a lot of useful things possible with dd. Some examples.

11.7.2. Create a CDROM image

The easiest way to create a **.ISO file** from any CD. The if switch means Input File, of is the Output File. Any good tool can burn a copy of the CD with this .ISO file.

```
dd if=/dev/cdrom of=/path/to/cdrom.ISO
```

11.7.3. Create a floppy image

A little outdated maybe, but just in case : make an image file from a 1.44MB floppy. Blocksize is defined by bs, and count contains the number of blocks to copy.

```
dd if=/dev/floppy of=/path/to/floppy.img bs=1024 count=1440
```

11.7.4. Copy the master boot record

Use dd to copy the **MBR** (Master Boot Record) of hard disk /dev/hda to a file.

```
dd if=/dev/hda of=/MBR.img bs=512 count=1
```

11.7.5. Copy files

This example shows how dd can copy files. Copy the file summer.txt to copy_of_summer.txt .

```
dd if=~/summer.txt of=~/copy_of_summer.txt
```

11.7.6. Image disks or partitions

And who needs ghost when dd can create a (compressed) image of a partition.

```
dd if=/dev/hdb2 of=/image_of_hdb2.IMG
dd if=/dev/hdb2 | gzip > /image_of_hdb2.IMG.gz
```

11.7.7. Create files of a certain size

dd can be used to create a file of any size. The first example creates a one MEBIbyte file, the second a one MEGAbyte file.

```
dd if=/dev/zero of=file1MB count=1024 bs=1024
dd if=/dev/zero of=file1MB count=1000 bs=1024
```

11.7.8. CDROM server example

And there are of course endless combinations with ssh and bzip2. This example puts a bzip2 backup of a cdrom on a remote server.

```
dd if=/dev/cdrom |bzip2|ssh user@host "cat - > /backups/cd/cdrom.iso.bz2"
```

11.8. split

The **split** command is useful to split files into smaller files. This can be useful to fit the file onto multiple instances of a medium too small to contain the complete file. In the example below, a file of size 5000 bytes is split into three smaller files, with maximum 2000 bytes each.

```
paul@laika:~/test$ ls -l
total 8
-rw-r--r-- 1 paul paul 5000 2007-09-09 20:46 bigfile1
paul@laika:~/test$ split -b 2000 bigfile1 splitfile.
paul@laika:~/test$ ls -l
total 20
-rw-r--r-- 1 paul paul 5000 2007-09-09 20:46 bigfile1
-rw-r--r-- 1 paul paul 2000 2007-09-09 20:47 splitfile.aa
-rw-r--r-- 1 paul paul 2000 2007-09-09 20:47 splitfile.ab
-rw-r--r-- 1 paul paul 1000 2007-09-09 20:47 splitfile.ac
```

11.9. Practice backup

!! Careful with tar options and the position of the backup file, mistakes can destroy your system!!

1. Create a directory (or partition if you like) for backups. Link (or mount) it under /mnt/backup.

2a. Use tar to backup /etc in /mnt/backup/etc_date.tgz, the backup must be gzipped. (Replace date with the current date)

2b. Use tar to backup /bin to /mnt/backup/bin_date.tar.bz2, the backup must be bzip2'd.

2c. Choose a file in /etc and /bin and verify with tar that the file is indeed backed up.

2d. Extract those two files to your home directory.

3a. Create a backup directory for your neighbour, make it accessible under /mnt/neighbourName

3b. Combine ssh and tar to put a backup of your /boot on your neighbours computer in /mnt/YourName

4a. Combine find and cpio to create a cpio archive of /etc.

4b. Choose a file in /etc and restore it from the cpio archive into your home directory.

5. Use dd and ssh to put a backup of the master boot record on your neighbours computer.

6. (On the real computer) Create and mount an ISO image of the ubuntu cdrom.

7. Combine dd and gzip to create a 'ghost' image of one of your partitions on another partition.

8. Use dd to create a five megabyte file in ~/testsplit and name it biggest. Then split this file in smaller two megabyte parts.

```
mkdir testsplit

dd if=/dev/zero of=~/testsplit/biggest count=5000 bs=1024

split -b 2000000 biggest parts
```

Chapter 12. Performance Monitoring

12.1. About Monitoring

Monitoring means obtaining information about the utilization of memory, CPU power, bandwidth and storage. You should start monitoring your system as soon as possible, to be able to create a **baseline**. Make sure that you get to know your system. *Boys, just give your computer a girls name and get to know her.* The baseline is important, it allows you to see a steady growth in CPU utilization or a steady decline in free disk space. It will allow you to plan for scaling up or scaling out.

Let us look at some tools that go beyond **ps fax**, **df -h**, **lspci**, **fdisk -l** and **du -sh**.

12.2. top

To start monitoring, you can use **top**. This tool will monitor Memory, CPU and running processes. Top will automatically refresh. Inside top you can use many commands, like **k** to kill processes, or **t** and **m** to toggle displaying task and memory information, or the number **1** to have one line per cpu, or one summary line for all cpu's.

```
top - 12:23:16 up 2 days,  4:01, 2 users, load average: 0.00, 0.00, 0.00
Tasks:  61 total,   1 running,  60 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.3% us,  0.5% sy, 0.0% ni, 98.9% id, 0.2% wa, 0.0% hi, 0.0% si
Mem:    255972k total,   240952k used,    15020k free,    59024k buffers
Swap:   524280k total,      144k used,   524136k free,   112356k cached

PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1 root      16   0  2816  560  480 S  0.0  0.2   0:00.91 init
 2 root      34  19     0    0    0 S  0.0  0.0   0:00.01 ksoftirqd/0
 3 root       5 -10     0    0    0 S  0.0  0.0   0:00.57 events/0
 4 root       5 -10     0    0    0 S  0.0  0.0   0:00.00 khelper
 5 root      15 -10     0    0    0 S  0.0  0.0   0:00.00 kacpid
16 root       5 -10     0    0    0 S  0.0  0.0   0:00.08 kblockd/0
26 root      15   0     0    0    0 S  0.0  0.0   0:02.86 pdflush
...
```

You can customize top to display the columns of your choice, or to display only the processes that you find interesting.

```
[paul@RHELv4u3 ~]$ top p 3456 p 8732 p 9654
```

12.3. free

The **free** command is common on Linux to monitor free memory. You can use free to display information every x seconds, but the output is not ideal.

```
[paul@RHELv4u3 gen]$ free -om -s 10
total       used       free     shared    buffers     cached
Mem:        249        222         27          0         50        109
Swap:       511          0        511

total       used       free     shared    buffers     cached
Mem:        249        222         27          0         50        109
Swap:       511          0        511

[paul@RHELv4u3 gen]$
```

12.4. watch

It might be more interesting to combine free with the **watch** program. This program can also run commands with a delay, and can highlight changes (with the -d switch).

```
[paul@RHELv4u3 ~]$ watch -d -n 3 free -om
...
Every 3.0s: free -om                            Sat Jan 27 12:13:03 2007

total       used       free     shared    buffers     cached
Mem:           249        230         19          0         56        109
Swap:          511          0        511
```

12.5. vmstat

To monitor CPU, disk and memory statistics in one line there is **vmstat**. The screenshot below shows vmstat running every two seconds 100 times (or until the Ctrl-C). Below the r, you see the number of processes waiting for the CPU, sleeping processes go below b. Swap usage (swpd) stayed constant at 144 kilobytes, free memory dropped from 16.7MB to 12.9MB. See man vmstat for the rest

```
[paul@RHELv4u3 ~]$ vmstat 2 100
procs -----------memory---------- --swap-- ---io--- --system-- ---cpu----
r  b   swpd   free   buff  cache   si   so   bi    bo   in    cs us sy id wa
0  0    144  16708  58212 111612    0    0    3     4   75    62  0  1 99  0
0  0    144  16708  58212 111612    0    0    0     0  976    22  0  0 100 0
0  0    144  16708  58212 111612    0    0    0     0  958    14  0  1 99  0
1  0    144  16528  58212 111612    0    0    0    18 1432  7417  1 32 66  0
1  0    144  16468  58212 111612    0    0    0     0 2910 20048  4 95  1  0
1  0    144  16408  58212 111612    0    0    0     0 3210 19509  4 97  0  0
1  0    144  15568  58816 111612    0    0  300  1632 2423 10189  2 62  0 36
0  1    144  13648  60324 111612    0    0  754     0 1910  2843  1 27  0 72
0  0    144  12928  60948 111612    0    0  312   418 1346  1258  0 14 57 29
0  0    144  12928  60948 111612    0    0    0     0  977    19  0  0 100 0
0  0    144  12988  60948 111612    0    0    0     0  977    15  0  0 100 0
0  0    144  12988  60948 111612    0    0    0     0  978    18  0  0 100 0

[paul@RHELv4u3 ~]$
```

12.6. iostat

The **iostat** tool can display disk and cpu statistics. The -d switch below makes iostat only display disk information (500 times every two seconds). The first block displays statistics since the last reboot.

```
[paul@RHELv4u3 ~]$ iostat -d 2 500
Linux 2.6.9-34.EL (RHELv4u3.localdomain)        01/27/2007

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
hdc               0.00         0.01         0.00       1080          0
sda               0.52         5.07         7.78     941798    1445148
sda1              0.00         0.01         0.00        968          4
sda2              1.13         5.06         7.78     939862    1445144
dm-0              1.13         5.05         7.77     939034    1444856
dm-1              0.00         0.00         0.00        360        288

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
hdc               0.00         0.00         0.00          0          0
sda               0.00         0.00         0.00          0          0
sda1              0.00         0.00         0.00          0          0
sda2              0.00         0.00         0.00          0          0
dm-0              0.00         0.00         0.00          0          0
dm-1              0.00         0.00         0.00          0          0
...
[paul@RHELv4u3 ~]$
```

You can have more statistics using **iostat -d -x**, or display only cpu statistics with **iostat -c**.

```
[paul@RHELv4u3 ~]$ iostat -c 5 500
Linux 2.6.9-34.EL (RHELv4u3.localdomain)        01/27/2007

avg-cpu:  %user   %nice    %sys %iowait   %idle
0.31    0.02    0.52    0.23   98.92

avg-cpu:  %user   %nice    %sys %iowait   %idle
0.62    0.00   52.16   47.23    0.00

avg-cpu:  %user   %nice    %sys %iowait   %idle
2.92    0.00   36.95   60.13    0.00

avg-cpu:  %user   %nice    %sys %iowait   %idle
0.63    0.00   36.63   62.32    0.42

avg-cpu:  %user   %nice    %sys %iowait   %idle
0.00    0.00    0.20    0.20   99.59

 [paul@RHELv4u3 ~]$
```

12.7. mpstat

On multi-processor machines, **mpstat** can display statistics for all, or for a selected cpu.

```
paul@laika:~$ mpstat -P ALL
Linux 2.6.20-3-generic (laika)  02/09/2007

CPU %user  %nice   %sys %iowait   %irq   %soft  %steal   %idle   intr/s
all  1.77   0.03   1.37    1.03   0.02    0.39    0.00   95.40  1304.91
  0  1.73   0.02   1.47    1.93   0.04    0.77    0.00   94.04  1304.91
  1  1.81   0.03   1.27    0.13   0.00    0.00    0.00   96.76     0.00
paul@laika:~$
```

12.8. sadc and sar

The **sadc** tool writes system utilization data to **/var/log/sa/sa??**, where ?? is replaced with the current day of the month. By default, cron runs the **sal** script every 10 minutes, the sal script runs sadc for one second. Just before midnight every day, cron runs the **sa2** script, which in turn invokes **sar**. The sar tool will read the daily data generated by sadc and put it in /var/log/sa/sar??. These **sar reports** contain a lot of statistics.

You can also use sar to display a portion of the statistics that were gathered. Like this example for cpu statistics.

```
[paul@RHELv4u3 sa]$ sar -u | head
Linux 2.6.9-34.EL (RHELv4u3.localdomain)        01/27/2007

12:00:01 AM       CPU     %user     %nice   %system   %iowait    %idle
12:10:01 AM       all      0.48      0.01      0.60      0.04    98.87
12:20:01 AM       all      0.49      0.01      0.60      0.06    98.84
12:30:01 AM       all      0.49      0.01      0.64      0.25    98.62
12:40:02 AM       all      0.44      0.01      0.62      0.07    98.86
12:50:01 AM       all      0.42      0.01      0.60      0.10    98.87
01:00:01 AM       all      0.47      0.01      0.65      0.08    98.80
01:10:01 AM       all      0.45      0.01      0.68      0.08    98.78
[paul@RHELv4u3 sa]$
```

There are other useful sar options, like **sar -I PROC** to display interrupt activity per interrupt and per CPU, or **sar -r** for memory related statistics. Check the manual page of sar for more.

12.9. ntop

The **ntop** tool is not present in default Red Hat installs. Once run, it will generate a very extensive analysis of network traffic in html on http://localhost:3000 .

12.10. iftop

The **iftop** tool will display bandwidth by socket statistics for a specific network device. Not available on default Red Hat servers.

```
             1.91Mb        3.81Mb         5.72Mb        7.63Mb   9.54Mb
-------------|-------------|--------------|-------------|--------|----
laika.local        => barry                        4.94Kb  6.65Kb  69.9Kb
                   <=                              7.41Kb  16.4Kb   766Kb
laika.local        => ik-in-f19.google.com            0b   1.58Kb  14.4Kb
                   <=                                 0b    292b   41.0Kb
laika.local        => ik-in-f99.google.com            0b     83b   4.01Kb
                   <=                                 0b     83b   39.8Kb
laika.local        => ug-in-f189.google.com           0b     42b    664b
                   <=                                 0b     42b    406b
laika.local        => 10.0.0.138                      0b      0b    149b
                   <=                                 0b      0b    256b
laika.local        => 224.0.0.251                     0b      0b     86b
                   <=                                 0b      0b      0b
laika.local        => ik-in-f83.google.com            0b      0b     39b
                   <=                                 0b      0b     21b
```

Chapter 13. User Quota's

13.1. About Disk Quotas

To limit the disk space used by user, you can set up **disk quotas**. This requires adding **usrquota** and/or **grpquota** to one or more of the file systems in **/etc/fstab**.

```
root@RHELv4u4:~# cat /etc/fstab | grep usrquota
/dev/VolGroup00/LogVol02     /home     ext3     usrquota,grpquota   0 0
```

Next you need to remount the file system.

```
root@RHELv4u4:~# mount -o remount /home
```

The next step is to build the **quota.user** and/or **quota.group** files. These files (called the **quota files**) contain the table of the disk usage on that file system. Use the **quotacheck** command to accomplish this.

```
root@RHELv4u4:~# quotacheck -cug /home
root@RHELv4u4:~# quotacheck -avug
```

The **-c** is for create, **u** for user quota, **g** for group, **a** for checking all quota enabled file systems in /etc/fstab and **v** for verbose information. The next step is to edit individual user quotas with **edquota** or set a general quota on the file system with **edquota -t**. The tool will enable you to put **hard** (this is the real limit) and **soft** (allows a grace period) limits on **blocks** and **inodes**. The **quota** command will verify that quota for a user is set. You can have a nice overview with **repquota**.

The final step (before your users start complaining about lack of disk space) is to enable quotas with **quotaon(1)**.

```
root@RHELv4u4:~# quotaon -vaug
```

Issue the **quotaoff** command to stop all complaints.

```
root@RHELv4u4:~# quotaoff -vaug
```

13.2. Practice Disk quotas

1. Implement disk quotas on one of your new partitions. Limit one of your users to 10 megabyte.

2. Test that they work by copying many files to the quota'd partition.

Chapter 14. VNC

14.1. About VNC

VNC can be configured in gnome or KDE using the **Remote Desktop Preferences**. **VNC** can be used to run your desktop on another computer, and you can also use it to see and take over the Desktop of another user. The last part can be useful for help desks to show users how to do things. VNC has the added advantage of being operating system independent, a lot of products (realvnc, tightvnc, xvnc, ...) use the same protocol on Solaris, Linux, BSD and more.

14.2. VNC Server

Starting the vnc server for the first time.

```
[root@RHELv4u3 conf]# rpm -qa | grep -i vnc
vnc-server-4.0-8.1
vnc-4.0-8.1
[root@RHELv4u3 conf]# vncserver :2

You will require a password to access your desktops.

Password:
Verify:
xauth:  creating new authority file /root/.Xauthority

New 'RHELv4u3.localdomain:2 (root)' desktop is RHELv4u3.localdomain:2

Creating default startup script /root/.vnc/xstartup
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/RHELv4u3.localdomain:2.log

[root@RHELv4u3 conf]#
```

14.3. VNC Client

You can now use the **vncviewer** from another machine to connect to your vnc server. It will default to a very simple graphical interface...

```
paul@laika:~$ vncviewer 192.168.1.49:2
VNC viewer version 3.3.7 - built Nov 20 2006 13:05:04
Copyright (C) 2002-2003 RealVNC Ltd.
Copyright (C) 1994-2000 AT&T Laboratories Cambridge.
See http://www.realvnc.com for information on VNC.
VNC server supports protocol version 3.8 (viewer 3.3)
Password:
VNC authentication succeeded
Desktop name "RHELv4u3.localdomain:2 (root)"
Connected to VNC server, using protocol version 3.3
...
```

If you don't like the simple twm window manager, you can comment out the last two lines of **~/.vnc/xstartup** and add a **gnome-session &** line to have vnc default to gnome instead.

```
[root@RHELv4u3 ~]# cat .vnc/xstartup
#!/bin/sh

# Uncomment the following two lines for normal desktop:
# unset SESSION_MANAGER
# exec /etc/X11/xinit/xinitrc

[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig -iconic &
# xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
# twm &
gnome-session &
[root@RHELv4u3 ~]#
```

Don't forget to restart your vnc server after changing this file.

```
[root@RHELv4u3 ~]# vncserver -kill :2
Killing Xvnc process ID 5785
[root@RHELv4u3 ~]# vncserver :2

New 'RHELv4u3.localdomain:2 (root)' desktop is RHELv4u3.localdomain:2

Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/RHELv4u3.localdomain:2.log
```

14.4. Practive VNC

1. Use VNC to connect from one machine to another.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

W

X

Y

Z