Tecmint.com
Linux Howto's Guide

Linux Foundation LFCS / LFCE Certification Preparation Guide

Linux Foundation Certified System Administrator (LFCS)

Linux Foundation Certified Engineer (LFCE)

DISCLAIMER:

This book covers the topics most likely to appear given the delivery technology, performance-based questions and time constraints of the exam. For example, you will note that we have not covered virtualization. If you have any questions about why we included a given topic and excluded another, feel free to reach us and we will be more than glad to clarify.

We hope you will enjoy reading this ebook as much as we enjoyed writing it and formatting it for distribution in PDF format. You will probably think of other ideas that can enrich this material. If so, feel free to drop us a note at one of our social network profiles:


http://twitter.com/tecmint


https://www.facebook.com/TecMint


https://plus.google.com/+Tecmint

In addition, if you find any typos or errors in this book, please let us know so that we can correct them and improve the material. Questions and other suggestions are appreciated as well – we look forward to hearing from you!

Contents

Chapter 1: Processing text streams in Linux 10
Using sed 10
Manipulating text files from the command line 12
Summary 13
Chapter 2: Edit files using vi/m 14
Launching vi 14
Understanding Vi modes 15
Vi commands 16
Options 17
Search and replace 17
Editing multiple files at a time 19
Temporary buffers 20
Summary 20
Chapter 3: Archiving and compression tools / File basic permissions and attributes 21
The tar utility 21
Using find to search for files 24
File permissions and basic attributes 25
Summary 27
Chapter 4: Partitioning storage devices, formatting filesystems, and configuring swap partitions 28
Managing MBR partitions with fdisk 28
Managing GPT partitions with gdisk 30
Formatting filesystems 31
Creating and using swap partitions 32
Summary 33
Chapter 5: Mounting and using filesystems 34
Mount options 35
Unmounting devices 36
Mounting networked filesystems 36
Mounting filesystems persistently 39

Examples 39
Summary 40

Chapter 6: RAIDs and backups 41

Assembling partitions as RAID devices 41
RAID Levels 44
Creating and managing system backups 46
Backing up your data 46
Summary 48

Chapter 7: Managing the startup process and related services 49

The boot process 49
Starting services 51
What about systemd? 55
Displaying information about the current status of a service 56
Starting or stopping services 56
Enabling or disabling a service to start during boot 57
Upstart 58
Summary 59

Chapter 8: User management, special attributes, and PAM 60

Adding user accounts 60
Deleting user accounts 63
Group management 63
Special permissions 63
SETUID 64
SETGID 64
STICKY BIT 65
Special file attributes 65
Accessing the root account and using sudo 66
PAM (Pluggable Authentication Modules) 68
Summary 70

Chapter 9: Package management 72

How package management systems work 72
Packaging Systems 72
High And Low-level Package Tools 72

Common usage of high-level tools 74
Installing a package from a repository 75
Removing a package 75
Displaying information about a package 75
Summary 76

Chapter 10: Terminals, shells, and filesystem troubleshooting 77

Basic shell scripting 78
Conditionals 79
FOR LOOPS 80
WHILE LOOPS 80
Putting it all together 80
Pinging a series of network or internet hosts for reply statistics 82
Filesystem troubleshooting 83
Summary 84

Chapter 11: Logical Volume Management 85

Creating physical volumes, volume groups, and logical volumes 85
Resizing logical volumes and extending volume groups 87
Mounting logical volumes on boot and on demand 89
Summary 90

Chapter 12: System documentation 91

Man pages 91
The --help option 92
Installed documentation in /usr/share/doc 93
GNU info 94
Summary 95

Chapter 13: The Grand Unified Bootloader (GRUB) 96

Introducing GRUB 96
Fixing GRUB issues 99
Summary 100

Chapter 14: Integrity and availability 101

Reporting processors statistics 101
Reporting processes 103

Setting resource limits on a per-user basis 104
Summary 106

Chapter 15: Kernel runtime parameters 107

Introducing the /proc filesystem 107
Other kernel runtime parameters explained 108
Summary 110

Chapter 16: SELinux and AppArmor 111

SELinux 111
AppArmor 115
Summary 118

Chapter 17: Access Control Lists (ACLs) and quotas 119

Checking file system compatibility with ACLs 119
Introducing ACLs 119
Setting ACLs 120
Disk quotas 122
Setting disk quotas 123
Summary 124

Chapter 18: Setting up network services 125

Installing a NFS server 125
Installing Apache 126
Installing Squid and SquidGuard 126
Postfix + Dovecot 126
Iptables 126
Configuring automatic start on boot 127
Summary 128

Chapter 19: The File Transfer Protocol (FTP) 129

Setting up a FTP server 129
Configuring the FTP server 129
Testing the FTP server 132
Summary 133

Chapter 20: Setting up a DNS server 135

Introducing name resolution 135

Installing and configuring a DNS server 136
Configuring zones 138
Testing the DNS server 140
Summary 143

Chapter 21: Configuring a database server 143

Installing and securing a MariaDB server 143
Configuring the database server 144
Checking the configuration 146
Summary 146

Chapter 22: Setting up a NFS server 148

Exporting network shares 148
Mounting exported network shares using autofs 150
Examining mounted file systems after starting the autofs daemon 150
Performing write tests in exported file systems 151
Summary 152

Chapter 23: The Apache web server 153

Configuring Apache 154
Serving pages in a standalone web server 154
Setting up name-based virtual hosts 156

Chapter 24: Squid 163

Configuring Squid – the basics 163
Verifying that a client can access the Internet 166
Restricting access by client 167
Configuring Squid – Fine tuning 168
Restricting access by user authentication 169
Using cache to speed up data transfer 170
Summary 172

Chapter 25: SquidGuard 173

Blacklists – the basics 173
Installing blacklists 175
Removing restrictions 177
Whitelisting specific domains and URLs 178
Summary 179

Chapter 26: Mail server 180
The process of sending and receiving email messages 180
Our testing environment 181
Adding email aliases 181
Configuring Postfix – the SMTP service 182
Restricting access to the SMTP server 183
Configuring Dovecot 184
Setting up a mail client and sending / receiving emails 185
Summary 188
Chapter 27: The firewall 189
The basics about iptables 189
Adding rules 191
Inserting, appending, and deleting rules 193
Summary 196
Chapter 28: Static and dynamic routing 197
IP and network device configuration 197
Summary 203
Chapter 29: Encrypted filesystems and swap space 204
Preparing a drive / partition / loop device for encryption 204
Testing for encryption support 204
Installing cryptsetup 205
Setting up an encrypted partition 205
Testing encryption 206
Encrypting the swap space for further security 207
Summary 208
Chapter 30: System usage, utilization, and troubleshooting 209
Storage space utilization 209
Memory and CPU utilization 211
A closer look at processes 213
Summary 217
Chapter 31: Setting up a network repository 218
Setting up a network repository server 218

Updating the repository 219
Configuring the client 221
Using the repository 222
Keeping the repository up-to-date 223
Summary 223

Chapter 32: Network performance, security, and troubleshooting 224

What services are running and why? 224
Investigating socket connections with ss 224
Protecting against port scanning with nmap 225
Reporting usage and performance on your network 227
Transferring files securely over the network 228
Summary 229

Chapter 1: Processing text streams in Linux

Linux treats the input to and the output from programs as streams (or sequences) of characters. To begin understanding redirection and pipes, we must first understand the three most important types of I/O (Input and Output) streams, which are in fact special files (by convention in UNIX and Linux, data streams and peripherals, or device files, are also treated as ordinary files).

The difference between > (redirection operator) and | (pipeline operator) is that while the first connects a command with a file, the latter connects the output of a command with another command:

```
command > file
command1 | command2
```

Since the redirection operator creates or overwrites files silently, we must use it with extreme caution, and never mistake it with a pipeline. One advantage of pipes on Linux and UNIX systems is that there is no intermediate file involved with a pipe - the stdout of the first command is not written to a file and then read by the second command.

For the following practice exercises we will use the poem "A happy child" (anonymous author):

```
gacanepa@debian:~/LFCS/lab1$ cat ahappychild.txt
My house is red - a little house;
A happy child am I.
I laugh and play the whole day long,
I hardly ever cry.
I have a tree, a green, green tree,
To shade me from the sun;
And under it I often sit,
When all my play is done.
gacanepa@debian:~/LFCS/lab1$
```

Using sed

The name sed is short for stream editor. For those unfamiliar with the term, a stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline).

The most basic (and popular) usage of sed is the substitution of characters. We will begin by changing every occurrence of the lowercase y to UPPERCASE Y and redirecting the output to ahappychild2.txt. The g flag indicates that sed should perform the substitution for all instances of term on every line of file. If this flag is omitted, sed will replace only the first occurrence of term on each line.

Basic syntax:

```
sed 's/term/replacement/flag' file
```

Our example:

```
sed 's/y/Y/g' ahappychild.txt > ahappychild2.txt
```

```
gacanepa@debian:~/LFCS/lab1$ sed 's/y/Y/g' ahappychild.txt > ahappychild2.txt
gacanepa@debian:~/LFCS/lab1$ cat ahappychild2.txt
MY house is red - a little house;
A happY child am I.
I laugh and plaY the whole daY long,
I hardlY ever crY.
I have a tree, a green, green tree,
To shade me from the sun;
And under it I often sit,
When all mY plaY is done.
gacanepa@debian:~/LFCS/lab1$
```

Should you want to search for or replace a special character (such as /, \, &) you need to escape it, in the term or replacement strings, with a backward slash.

For example, we will substitute the word and for an ampersand. At the same time, we will replace the word I with You when the first one is found at the beginning of a line.

```
sed 's/and/\&/g;s/^I/You/g' ahappychild.txt
```

```
gacanepa@debian:~/LFCS/lab1$ sed 's/and/\&/g;s/^I/You/g' ahappychild.txt
My house is red - a little house;
A happy child am I.
You laugh & play the whole day long,
You hardly ever cry.
You have a tree, a green, green tree,
To shade me from the sun;
And under it I often sit,
When all my play is done.
gacanepa@debian:~/LFCS/lab1$ ^C
gacanepa@debian:~/LFCS/lab1$
```

In the above command, a ^ (caret sign) is a well-known regular expression that is used to represent the beginning of a line.

As you can see, we can combine two or more substitution commands (and use regular expressions inside them) by separating them with a semicolon and enclosing the set inside single quotes.

Another use of sed is showing (or deleting) a chosen portion of a file. In the following example, we will display the first 5 lines of /var/log/messages from Jun 8:

```
sed -n '/^Jun  8/ p' /var/log/messages | sed -n 1,5p
```

Note that by default, sed prints every line. We can override this behavior with the -n option and then tell sed to print (indicated by p) only the part of the file (or the pipe) that matches the pattern (Jun 8 at the beginning of line in the first case and lines 1 through 5 inclusive in the second case).

Finally, it can be useful while inspecting scripts or configuration files to inspect the code itself and leave out comments. The following sed one-liner deletes (d) blank lines or those starting with # (the | character indicates a boolean OR between the two regular expressions):

```
sed '/^#\|^$/d' apache2.conf
```

```
root@debian:/etc/apache2# sed '/^#\|^$/d' apache2.conf
LockFile ${APACHE_LOCK_DIR}/accept.lock
PidFile ${APACHE_PID_FILE}
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5
<IfModule mpm_prefork_module>
    StartServers          5
    MinSpareServers       5
    MaxSpareServers      10
    MaxClients          150
    MaxRequestsPerChild   0
</IfModule>
<IfModule mpm_worker_module>
    StartServers          2
    MinSpareThreads      25
    MaxSpareThreads      75
```

Manipulating text files from the command line

The **uniq** command allows us to report or remove duplicate lines in a file, writing to stdout by default. We must note that **uniq** does not detect repeated lines unless they are adjacent. Thus, **uniq** is commonly used along with a preceding **sort** (which is used to sort lines of text files). By default, **sort** takes the first field (separated by spaces) as key field. To specify a different key field, we need to use the **-k** option.

grep searches text files or (command output) for the occurrence of a specified regular expression and outputs any line containing a match to standard output.

The **tr** command can be used to translate (change) or delete characters from stdin, and write the result to stdout.

The **cut** command extracts portions of input lines (from stdin or files) and displays the result on standard output, based on number of bytes (**-b** option), characters (**-c**), or fields (**-f**). In this last case (based on fields), the default field separator is a tab, but a different delimiter can be specified by using **-d**.

For example, we will create a text stream consisting of the first and third non-blank files of the output of the **last** command. We will use **grep** as a first filter to check for sessions of user gacanepa, then squeeze delimiters to

only one space (**tr -s ' '**). Next, we'll extract the first and third fields with cut, and finally sort by the second field (IP addresses in this case) showing unique addresses:

```
Last | grep gacanepa | tr -s ' ' | cut -d' ' -f1,3 | sort -k2 | uniq
```

```
gacanepa@debian:~/LFCS/lab1$ last | grep gacanepa | tr -s ' ' | cut -d' ' -f1,3 | sort -k2 | uniq
gacanepa [redacted].28
gacanepa [redacted].29
gacanepa [redacted].5
gacanepa 192.168.0.101
gacanepa 192.168.0.102
gacanepa 192.168.0.103
gacanepa 192.168.0.105
gacanepa 192.168.0.106
```

The above command shows how multiple commands and pipes can be combined so as to obtain filtered data according to our desires. Feel free to also run it by parts, to help you see the output that is pipelined from one command to the next (this can be a great learning experience, by the way!).

Summary

Although this example (along with the rest of the examples in the current tutorial) may not seem very useful at first sight, they are a nice starting point to begin experimenting with commands that are used to create, edit, and manipulate files from the Linux command line.

Chapter 2: Edit files using vi/m

Vi was the first full-screen text editor written for Unix. Although it was intended to be small and simple, it can be a bit challenging for people used exclusively to GUI text editors, such as NotePad++, or gedit, to name a few examples.

To use Vi, we must first understand the 3 modes in which this powerful program operates, in order to begin learning later about the its powerful text-editing procedures.

Please note that most modern Linux distributions ship with a variant of vi known as vim ("Vi improved"), which supports more features than the original vi does. For that reason, throughout this tutorial we will use vi and vim interchangeably.

If your distribution does not have vim installed, you can install it as follows:

Ubuntu and derivatives:

```
aptitude update && aptitude install vim
```

Red Hat-based distributions:

```
yum update && yum install vim
```

openSUSE:

```
zypper update && zypper install vim
```

Why should I want to learn vi?

There are at least 2 good reasons to learn vi:

1) vi is always available (no matter what distribution you're using) since it is required by POSIX.

2) vi does not consume a considerable amount of system resources and allows us to perform any imaginable tasks without lifting our fingers from the keyboard.

In addition, vi has a very extensive built-in manual, which can be launched using the **:help** command right after the program is started. This built-in manual contains more information than vi/m's man page.

Launching vi

To launch **vi**, type **vi** in your command prompt

then press i to enter Insert mode, and you can start typing.

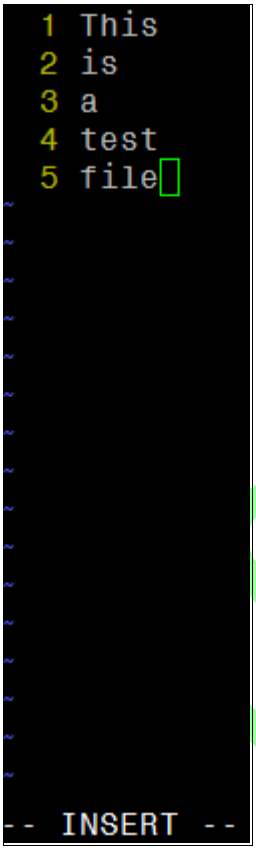
Another way to launch vi/m is

```
vi filename
```

which will open a new buffer (more on buffers later) named filename, which you can later save to disk.

Understanding Vi modes

1. In command mode, vi allows the user to navigate around the file and enter vi commands, which are brief, case-sensitive combinations of one or more letters. Almost all of them can be prefixed with a number to repeat the command that number of times. For example, yy (or Y) copies the entire current line, whereas 3yy (or 3Y) copies the entire current line along with the two next lines (3 lines in total). We can always enter command mode (regardless of the mode we're working on) by pressing the Esc key. The fact that in command mode the keyboard keys are interpreted as commands instead of text tends to be confusing to beginners.
2. In ex mode, we can manipulate files (including saving a current file and running outside programs). To enter this mode, we must type a colon (:) from command mode, directly followed by the name of the ex-mode command that needs to be used. After that, vi returns automatically to command mode.
3. In insert mode (the letter i is commonly used to enter this mode), we simply enter text. Most keystrokes result in text appearing on the screen (one important exception is the Esc key, which exits insert mode and returns to command mode):



Vi commands

The following table shows a list of commonly used vi commands. File edition commands can be enforced by appending the exclamation sign to the command (for example, :q! enforces quitting without saving).

Key command	Description
h or left arrow	Go one character to the left
j or down arrow	Go down one line
k or up arrow	Go up one line
l (lowercase L) or right arrow	Go one character to the right
H	Go to the top of the screen
L	Go to the bottom of the screen
G	Go to the end of the file
w	Move one word to the right
b	Move one word to the left
0 (zero)	Go to the beginning of the current line
^	Go to the first nonblank character on the current line
$	Go to the end of the current line
Ctrl-B	Go back one screen
Ctrl-F	Go forward one screen
i	Insert at the current cursor position
I (uppercase i)	Insert at the beginning of the current line
J (uppercase j)	Join current line with the next one (move next line up)
a	Append after the current cursor position
o (lowercase O)	Creates a blank line after the current line
O (uppercase o)	Creates a blank line before the current line
r	Replace the character at the current cursor position
R	Overwrite at the current cursor position
x	Delete the character at the current cursor position
X	Delete the character immediately before (to the left) of the current cursor position
dd	Cut (for later pasting) the entire current line
D	Cut from the current cursor position to the end of the line (this command is equivalent to d$)
yX	Give a movement command X, copy (yank) the appropriate number of characters, words, or lines from the current cursor position
yy or Y	Yank (copy) the entire current line
p	Paste after (next line) the current cursor position
P	Paste before (previous line) the current cursor position
. (period)	Repeat the last command
u	Undo the last command
U	Undo the last command in the last line. This will work as long as the cursor is still on the line.
n	Find the next match in a search
N	Find the previous match in a search
:n	Next file; when multiple files are specified for editing, this commands loads the next file.

:e file	Load file in place of the current file.
:r file	Insert the contents of file after (next line) the current cursor position
:q	Quit without saving changes.
:w file	Write the current buffer to file. To append to an existing file, use :w >> file.
:wq	Write the contents of the current file and quit. Equivalent to x! and ZZ
:r! command	Execute command and insert output after (next line) the current cursor position.

Options

The following options can come in handy while running vim (we need to add them in our ~/.vimrc file):

```
echo set number >> ~/.vimrc
echo syntax on >> ~/.vimrc
echo set tabstop=4 >> ~/.vimrc
echo set autoindent >> ~/.vimrc
```

```
gacanepa@debian:~$ cat .vimrc
set number
syntax on
set tabstop=4
set autoindent
gacanepa@debian:~$
```

1) **set number** shows line numbers when vi opens an existing or a new file.

2) **syntax on** turns on syntax highlighting (for multiple file extensions) in order to make code and config files more readable.

3) **set tabstop=4** sets the tab size to 4 spaces (default value is 8)

4) **set autoindent** carries over previous indent to the next line

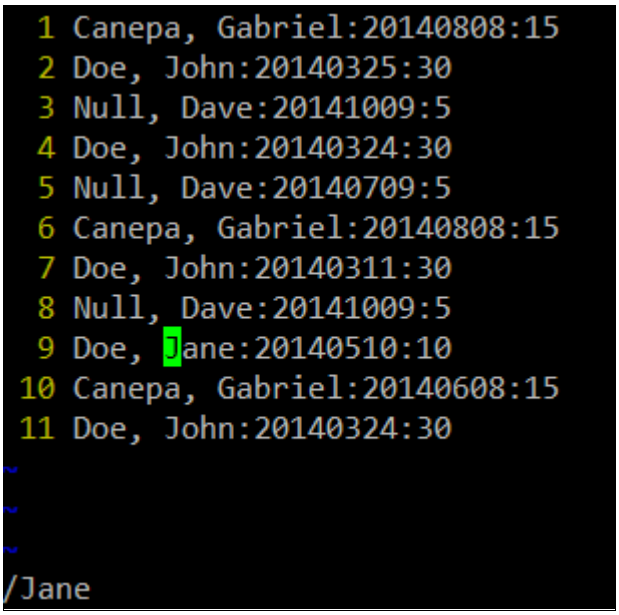
Search and replace

vi has the ability to move the cursor to a certain location (on a single line or over an entire file) based on searches. It can also perform text replacements with or without confirmation from the user.

a) Searching within a line: the f command searches a line and moves the cursor to the next occurrence of a specified character in the current line. For example, the command fh would move the cursor to the next instance of the letter h within the current line. Note that neither the letter f nor the character you're searching for will appear anywhere on your screen, but the character will be highlighted after you press Enter. For example, this is what I get after pressing f4 in command mode:

```
1 Canepa, Gabriel:20140808:15
2 Doe, John:20140325:30
3 Null, Dave:20141009:5
4 Doe, John:20140324:30
```

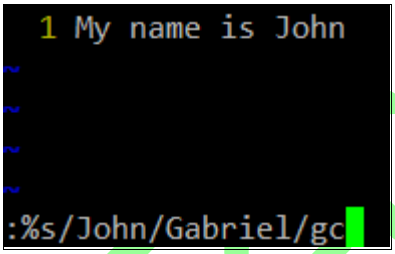
b) Searching an entire file: use the / command, followed by the word or phrase to be searched for. A search may be repeated using the previous search string with the n command, or the next one (using the N command). This is the result of typing /Jane in command mode:



vi uses a command (similar to sed's) to perform substitution operations over a range of lines or an entire file. To change the word "old" to "young" for the entire file, we must enter the following command:

```
:%s/old/young/g
```

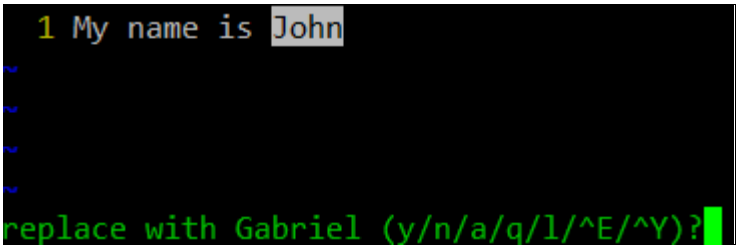
(notice the colon at the beginning of the command)



The colon (:) starts the ex command, s in this case (for substitution), % is a shortcut meaning from the first line to the last line (the range can also be specified as n,m which means "from line n to line m"), old is the search pattern, while young is the replacement text, and g indicates that the substitution should be performed on every occurrence of the search string in the file. Alternatively, a c can be added to the end of the command to ask for confirmation before performing any substitution:

```
:%s/old/young/gc
```

Before replacing the original text with the new one, vi/m will present us with the following message:



a) y: perform the substitution (yes)

b) n: skip this occurrence and go to the next one (no)

c) a: perform the substitution in this and all subsequent instances of the pattern.

d) q or Esc: quit substituting.

e) l (lowercase L): perform this substitution and quit (last)

f) Ctrl-e, Ctrl-y: Scroll down and up, respectively, to view the context of the proposed substitution.

Editing multiple files at a time

Let's type

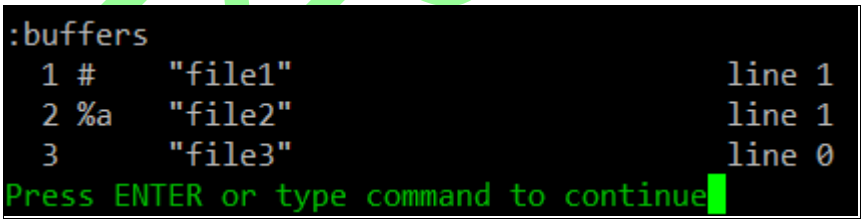
```
vim file1 file2 file3
```

in our command prompt.

First, vim will open file1. To switch to the next file (file2), we need to use the **:n** command. When we want to return to the previous file, **:N** will do the job.

In order to switch from file1 to file3:

a) the **:buffers** command will show a list of the file currently being edited:



b) The command **:buffer 3** (without the s at the end) will open file3 for editing.

In the image above, a pound sign (#) indicates that the file is currently open but in the background, while %a marks the file that is currently being edited. On the other hand, a blank space after the file number (3 in the above example) indicates that the file has not yet been opened.

Temporary buffers

To copy a couple of consecutive lines (let's say 4, for example) into a temporary buffer named a (not associated with a file) and place those lines in another part of the file later in the current vi section, we need to...

1) press the ESC key to be sure we are in vi Command mode.

2) place the cursor on the first line of the text we wish to copy.

3) type **"a4yy** to copy the current line, along with the 3 subsequent lines, into a buffer named a. We can continue editing our file - we do not need to insert the copied lines immediately.

4) When we reach the location for the copied lines, use **"a** before the p or P commands to insert the lines copied into the buffer named **a**:

4a) Type **"ap** to insert the lines copied into buffer a after the current line on which the cursor is resting.

4b) Type **"aP** to insert the lines copied into buffer a before the current line.

If we wish, we can repeat the above steps to insert the contents of buffer a in multiple places in our file. A temporary buffer, as the one in this section, is disposed when the current window is closed.

Summary

As we have seen, vi/m is a powerful and versatile text editor for the command line environment. As a system administrator you can do yourself a favor if you decide to learn how to use it.

Chapter 3: Archiving and compression tools / File basic permissions and attributes

A file archiving tool groups a set of files into a single standalone file that we can backup to several types of media, transfer across a network, or send via email. The most frequently used archiving utility in Linux is tar. When an archiving utility is used along with a compression tool, it allows to reduce the disk size that is needed to store the same files and information.

The tar utility

tar bundles a group of files together into a single archive (commonly called a tar file or tarball). The name originally stood for tape archiver, but we must note that we can use this tool to archive data to any kind of writeable media (not only to tapes). Tar is normally used with a compression tool such as gzip, bzip2, or xz to produce a compressed tarball.

Basic syntax:

```
tar [options] [pathname ...]
```

where … represents the expression used to specify which files should be acted upon.

Most commonly used tar commands:

Long option	Abbreviation	Description
--create	c	Creates a tar archive
--concatenate	A	Appends tar files to an archive
--append	r	Appends files to the end of an archive
--update	u	Appends files newer than copy in archive
--diff or --compare	d	Find differences between archive and file system
--file ARCHIVE	f	Use archive file or device ARCHIVE
--list	t	Lists the contents of a tarball
--extract or --get	x	Extracts files from an archive

Normally used operation modifiers:

Long option	Abbreviation	Description
--directory dir	C	Changes to directory dir before performing operations
--same-permissions	p	Preserves original permissions
--verbose	v	Lists all files read or extracted. When this flag is used along with --list, the file sizes, ownership, and time stamps are displayed.
--verify	W	Verifies the archive after writing it
--exclude file	---	Excludes file from the archive

--exclude=pattern	X	Exclude files, given as a PATTERN
--gzip or --gunzip	z	Processes an archive through gzip
--bzip2	j	Processes an archive through bzip2
--xz	J	Processes an archive through xz

Gzip is the oldest compression tool and provides the least compression, while bzip2 provides improved compression. In addition, xz is the newest but (usually) provides the best compression. This advantages of best compression come at a price: the time it takes to complete the operation, and system resources used during the process.

Normally, tar files compressed with these utilities have .gz, .bz2, or .xz extensions, respectively. In the following examples we will be using these files: file1, file2, file3, file4, and file5.

EXAMPLE 1: Group all the files in the current working directory and compress the resulting bundle with gzip, bzip, and xz (please note the use of a regular expression to specify which files should be included in the bundle - this is to prevent the archiving tool to group the tarballs created in previous steps)

```
tar czf myfiles.tar.gz file[0-9]
tar cjf myfiles.tar.bz2 file[0-9]
tar cJf myfile.tar.xz file[0-9]
```

```
gacanepa@debian:~/LFCS/lab3$ ls -lh | grep tar
-rw-r--r-- 1 gacanepa gacanepa  79K Oct 13 09:44 myfiles.tar.bz2
-rw-r--r-- 1 gacanepa gacanepa 101K Oct 13 09:43 myfiles.tar.gz
-rw-r--r-- 1 gacanepa gacanepa  84K Oct 13 09:44 myfiles.tar.xz
gacanepa@debian:~/LFCS/lab3$
```

EXAMPLE 2: List the contents of a tarball and display the same information as a long directory listing. Note that update or append operations cannot be applied to compressed files directly (if you need to update or append a file to a compressed tarball, you need to uncompress the tar file and update / append to it, then compress again).

```
tar tvf [tarball]
```

```
gacanepa@debian:~/LFCS/lab3$ tar tvf myfiles.tar.gz
-rw-r--r-- gacanepa/gacanepa 7986 2014-10-13 09:16 file1
-rw-r--r-- gacanepa/gacanepa 2232 2014-10-13 09:16 file2
-rw-r--r-- gacanepa/gacanepa 3214 2014-10-13 09:17 file3
-rw-r--r-- gacanepa/gacanepa 321072 2014-10-13 09:20 file4
-rw-r--r-- gacanepa/gacanepa  37271 2014-10-13 09:21 file5
gacanepa@debian:~/LFCS/lab3$
```

Run any of the following commands:

```
gzip -d myfiles.tar.gz
bzip2 -d myfiles.tar.bz2
xz -d myfiles.tar.xz
```

Then

```
tar --delete --file myfiles.tar file4
```

(deletes the file inside the tarball)

```
tar --update --file myfiles.tar file4
```

(adds the updated file)

and

1. `gzip myfiles.tar`, if you choose #1 above.
2. `bzip2 myfiles.tar`, if you choose #2.
3. `xz myfiles.tar`, if you choose #3.

Finally,

```
tar tvf [tarball] again
```

and compare the modification date and time of file4 with the same information as shown earlier.

EXAMPLE 3: Suppose you want to perform a backup of users' home directories. A good sysadmin practice would be (may also be specified by company policies) to exclude all video and audio files from backups.

Maybe your first approach would be to exclude from the backup all files with an .mp3 or .mp4 extension (or other extensions). What if you have a clever user who can change the extension to .txt or .bkp, your approach won't do you much good. In order to detect an audio or video file, you need to check its file type with file. The following shell script will do the job:

```
#!/bin/bash
# Pass the directory to backup as first argument.
DIR=$1
# Create the tarball and compress it. Exclude files with the MPEG string in its file type.
# -If the file type contains the string mpeg, $? (the exit status of the most recently
#  executed command) expands to 0, and the filename is redirected to the exclude option.
#  Otherwise, it expands to 1.
# -If $? equals 0, add the file to the list of files to be backed up.
tar X <(for i in $DIR/*; do file $i | grep -i mpeg; if [ $? -eq 0 ]; then echo $i; fi;done) -
```


```
#!/bin/bash
# Pass the directory to backup as first argument.
DIR=$1
# Create the tarball and compress it. Exclude files with the MPEG string in its
file type.
# -If the file type contains the string mpeg, $? (the exit status of the most
recently executed command) expands to 0, and the filename is redirected to the
exclude option. Otherwise, it expands to 1.
```

```
# -If $? equals 0, add the file to the list of files to be backed up.
tar X <(for i in $DIR/*; do file $i | grep -i mpeg; if [ $? -eq 0 ]; then echo
$i; fi;done) -cjf backupfile.tar.bz2 $DIR/*
```

EXAMPLE 4: You can then restore the backup to the original user's home directory (user_restore in this example), preserving permissions, with the following command:

```
tar xjf backupfile.tar.bz2 --directory user_restore --same-permissions
```

```
root@debian:/home/gacanepa/LFCS/lab3# tar xjf backupfile.tar.bz2 --directory user_restore --same
root@debian:/home/gacanepa/LFCS/lab3# ls user_restore/*
user_restore/backupfile.bz2  user_restore/file1  user_restore/file3  user_restore/file5
user_restore/backup.sh       user_restore/file2  user_restore/file4  user_restore/myfiles.tar.bz

user_restore/aux:
file1  file2  file3  file4  file5
root@debian:/home/gacanepa/LFCS/lab3#
```

Using find to search for files

The find command is used to search recursively through directory trees for files or directories that match certain characteristics, and can then either print the matching files or directories or perform other operations on the matches.

Normally, we will search by name, owner, group, type, permissions, date, and size.

Basic syntax:

```
find [directory_to_search] [expression]
```

EXAMPLE 1: Find all files (-f) in the current directory (.) and 2 subdirectories below (-maxdepth 3 includes the current working directory and 2 levels down) whose size (-size) is greater than 2 MB

```
find . -maxdepth 3 -type f -size +2M
```

```
root@debian:/home/gacanepa# find . -maxdepth 3 -type f -size +2M
./LFCS/lab3/SAPProductSuite.mp4
./LFCS/lab3/StayWithMeTonight.mp3
./Books/Wiley.Linux.Command.Line.and.Shell.Scripting.Bible.May.2008.pdf
./Books/Head.First.C.pdf
./Books/libre_m2_baja.pdf
root@debian:/home/gacanepa#
```

EXAMPLE 2: Files with 777 permissions are sometimes considered an open door to external attackers. Either way, it is not safe to let anyone do anything they please with files. We will take a rather aggressive approach and delete them! ('{}' + is used to "collect" the results of the search)

```
find /home/user -perm 777 -exec rm '{}' +
```

```
root@debian:/home/gacanepa# ls -l | grep sed
-rwxrwxrwx 1 gacanepa gacanepa 13761 Oct  8 09:04 sed.pdf
root@debian:/home/gacanepa# find . -perm 777 -exec rm '{}' +
root@debian:/home/gacanepa# ls -l | grep sed
root@debian:/home/gacanepa#
```

EXAMPLE 3: Search for configuration files in /etc that have been accessed (-atime) or modified (-mtime) more (+180) or less (-180) than 6 months ago or exactly 6 months ago (180)

Modify the following command as per the example below:

```
find /etc -iname "*.conf" -mtime -180 -print
```

File permissions and basic attributes

The first 10 characters in the output of ls -l are the file attributes. The first of these characters is used to indicate the file type:

- -: a regular file
- -d: a directory
- -l: a symbolic link
- -c: a character device (which treats data as a stream of bytes, i.e. a terminal)
- -b: a block device (which handles data in blocks, i.e. storage devices)

The next nine characters of the file attributes are called the file mode and represent the read (r), write(w), and execute (x) permissions of the file's owner, the file's group owner, and the rest of the users (commonly referred to as "the world").

Whereas the read permission on a file allows the same to be opened and read, the same permission on a directory allows its contents to be listed if the execute permission is also set. In addition, the execute permission in a file allows it to be handled as a program and run, while in a directory it allows the same to be cd'ed into it.

File permissions are changed with the chmod command, whose basic syntax is as follows:

```
chmod [new_mode] file
```

where `new_mode` is either an octal number or an expression that specifies the new permissions.

The octal number can be converted from its binary equivalent, which is calculated from the desired file permissions for the owner, the group, and the world, as follows:

The presence of a certain permission equals a power of 2 (r=22, w=21, x=20), while its absence equates to 0. For example:

r	w	x	r	-	-	r	-	-
4	2	1	4	0	0	4	0	0
4+2+1=7			4+0+0=4			4+0+0=4		

To set the file's permissions as above in octal form, type:

```
chmod 744 myfile
```

You can also set a file's mode using an expression that indicates the owner's rights with the letter u, the group owner's rights with the letter g, and the rest with o. All of these rights can be represented at the same time with the letter a. Permissions are granted (or revoked) with the + or - signs, respectively.

EXAMPLE 1: Revoke execute permission for backup.sh to all users

```
chmod a-x backup.sh
```

EXAMPLE 2: Grant read and write permissions for myfile to the owner and group owner

```
chmod 770 myfile
```

In time, and with practice, you will be able to decide which method to change a file mode works best for you in each case.

A long directory listing also shows the file's owner and its group owner (which serve as a rudimentary yet effective access control to files in a system):

```
root@debian:~# ls -l
total 1808
-rw-r--r-- 1 root root      48 Oct  6 13:51 email_body.txt
drwx------ 2 root root    4096 Oct  3 22:06 PDF
-rw-r--r-- 1 root root    1105 Oct 11 00:46 reporte.html
-rw------- 1 root root  386504 Oct  6 14:48 sent
-rw-r--r-- 1 root root       0 Jun  9 17:58 TestFile
-rw-r--r-- 1 root root 1298432 May  6  2013 upgrade-wheezystep.script
-rw-r--r-- 1 root root  138674 May  6  2013 upgrade-wheezystep.time
root@debian:~#
```

File ownership is changed with the chown command. The owner and the group owner can be changed at the same time or separately. Its basic syntax is as follows:

```
chown user:group file
```

where at least user or group need to be present.

Alternatively, you can change the group owner of **file** to **new_group** with

```
chgrp new_group file
```

Example 1: Change the owner of **sent** to user gacanepa

```
chown gacanepa sent
```

Example 2: Change the owner and group of **TestFile** to user and group **gacanepa**:

```
chown gacanepa:gacanepa TestFile
```

Example 3: Change the group owner of **email_body.txt** to **gacanepa**:

```
chown :gacanepa email_body.txt
```

Summary

In this chapter we have explained how to use group and compress files in Linux, and discussed basic file permissions and attributes. These are essential skills that every system administrator must possess and are central to passing the exam.

Chapter 4: Partitioning storage devices, formatting filesystems, and configuring swap partitions

Partitioning is a means to divide a single hard drive into one or more parts or “slices” called partitions. A partition is a section on a drive that is treated as an independent disk and which contains a single type of file system, whereas a partition table is an index that relates those physical sections of the hard drive to partition identifications.

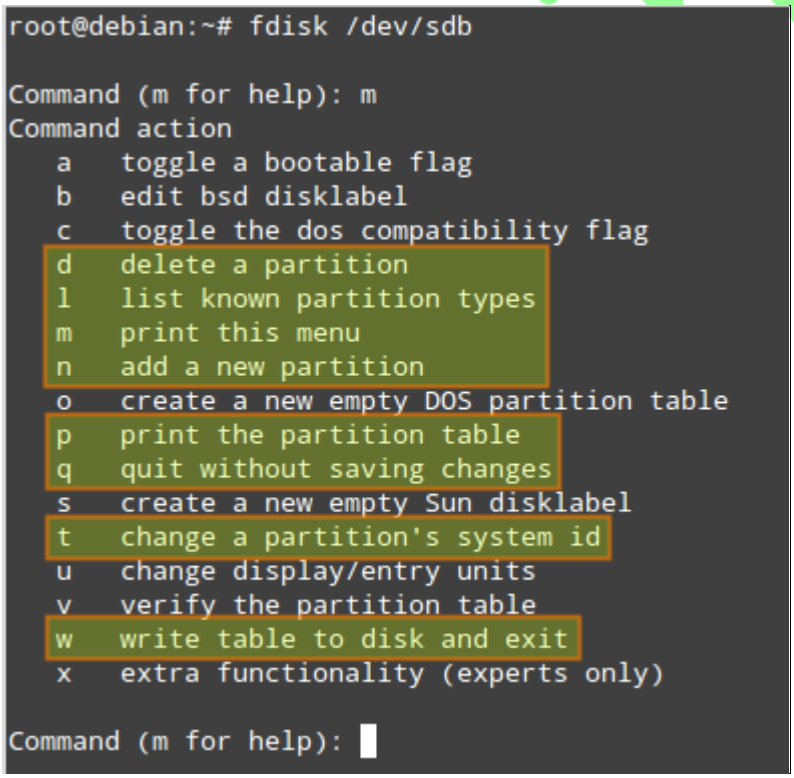
In Linux, the traditional tool for managing MBR partitions (up to ~2009) in IBM PC compatible systems is fdisk. For GPT partitions (~2010 and later) we will use gdisk. Each of these tools can be invoked by typing its name followed by a device name (such as /dev/sdb).

Managing MBR partitions with fdisk

We will cover **fdisk** first:

```
fdisk /dev/sdb
```

A prompt appears asking for the next operation. If you are unsure, you can press the m key to display the help contents:



In the above image, the most frequently used options are highlighted:

At any moment, you can press p to display the current partition table:

```
Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders, total 16777216 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xd8cf056d

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048    10487807     5242880   83  Linux
/dev/sdb2        10487808    16777215     3144704   83  Linux
```

The Id column shows the partition type (or partition id) that has been assigned by fdisk to the partition. A partition type serves as an indicator of the file system the partition contains or, in simple words, the way data will be accessed in that partition. Please note that a comprehensive study of each partition type is out of the scope of this tutorial - as this book is focused on the LFCS exam, which is performance-based.

- You can list all the partition types that can be managed by fdisk by pressing the l option (lowercase L).
- Press d to delete an existing partition. If more than one partition is found in the drive, you will be asked which one should be deleted. Enter the corresponding number, and then press w (write modifications to partition table) to apply changes. In the following example, we will delete /dev/sdb2, and then print (p) the partition table to verify the modifications:

```
Command (m for help): d
Partition number (1-4): 2

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders, total 16777216 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xd8cf056d

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048    10487807     5242880   83  Linux

Command (m for help):
```

- If the partition Id that fdisk chose is not the right one for our setup, we can press **t** to change it:

```
Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 7
Changed system type of partition 1 to 7 (HPFS/NTFS/exFAT)

Command (m for help): p

Disk /dev/sdb: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders, total 16777216 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xd8cf056d

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048    10487807     5242880    7  HPFS/NTFS/exFAT

Command (m for help):
```

When you're done setting up the partitions, press w to commit the changes to disk:

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

Calling ioctl() to re-read partition table.
Syncing disks.
```

Managing GPT partitions with gdisk

In the following example, we will use /dev/sdb:

```
gdisk /dev/sdb
```

We must note that gdisk can be used either to create MBR or GPT partitions:

```
root@debian:~# gdisk /dev/sdb
GPT fdisk (gdisk) version 0.8.5

Partition table scan:
  MBR: MBR only
  BSD: not present
  APM: not present
  GPT: present

Found valid MBR and GPT. Which do you want to use?
 1 - MBR
 2 - GPT
 3 - Create blank GPT

Your answer: 2
Using GPT and creating fresh protective MBR.
```

The advantage of using GPT partitioning is that we can create up to 128 partitions in the same disk whose size can be up to the order of petabytes, whereas the maximum size for MBR partitions is 2 TB.

Note that most of the options in fdisk are the same in gdisk. For that reason, we will not go into detail about them.

Formatting filesystems

Once we have created all the necessary partitions, we must create filesystems. To find out the list of filesystems supported in your system, run:

```
ls /sbin/mk*
```

```
root@debian:~# ls /sbin/mk*
/sbin/mke2fs    /sbin/mkfs.cramfs  /sbin/mkfs.ext4     /sbin/mkhomedir_helper
/sbin/mkfs      /sbin/mkfs.ext2    /sbin/mkfs.ext4dev  /sbin/mkswap
/sbin/mkfs.bfs  /sbin/mkfs.ext3    /sbin/mkfs.minix
root@debian:~#
```

The type of filesystem that you should choose depends on your requirements. You should consider the pros and and cons of each filesystem and its own set of features. Two important attributes to look for in a filesystem are:

- Journaling support, which allows for faster data recovery in the event of a system crash.
- Security Enhanced Linux (SELinux) support, as per the project wiki, “a security enhancement to Linux which allows users and administrators more control over access control”.

In our next example, we will create an ext4 filesystem (supports both journaling and SELinux) labeled Tecmint on /dev/sdb1, using mkfs, whose basic syntax is:

```
mkfs -t [filesystem] -L [label] device
```

or

```
mkfs.[filesystem] -L [label] device
```

```
root@debian:~# mkfs.ext4 -L Tecmint /dev/sdb1
mke2fs 1.42.5 (29-Jul-2012)
Filesystem label=Tecmint
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
524288 inodes, 2096891 blocks
104844 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2147483648
64 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

root@debian:~#
```

Creating and using swap partitions

Swap partitions are necessary if we need our Linux system to have access to virtual memory, which is a section of the hard disk designated for use as memory when the main system memory (RAM) is all in use. For that reason, a swap partition may not be needed on systems with enough RAM to meet all its requirements; however, even in that case it's up to the system administrator to decide whether to use a swap partition or not.

A simple rule of thumb to decide the size of a swap partition is as follows:

Swap should usually equal 2x physical RAM for up to 2 GB of physical RAM, and then an additional 1x physical RAM for any amount above 2 GB, but never less than 32 MB.

So, if:

M = Amount of RAM in GB, and S = Amount of swap in GB, then

If M < 2

S = M *2

Else

S = M + 2

Remember this is just a formula and that only you, as a sysadmin, have the final word as to the use and size of a swap partition.

To configure a swap partition, create a regular partition as demonstrated earlier with the desired size.

Next, we need to add the following entry to the /etc/fstab file (X can be either b or c):

```
/dev/sdX1 swap swap sw 0 0
```

Finally, let's format and enable the swap partition:

```
mkswap /dev/sdX1
swapon -v /dev/sdX1
```

To display a snapshot of the swap partition(s):

```
cat /proc/swaps
```

To disable the swap partition:

```
swapoff /dev/sdX1
```

Summary

Creating partitions (including swap) and formatting filesystems are crucial in your road to becoming a system administrator and passing the LFCS exam. I hope that the tips provided in the present chapter will help you to achieve that goal.

Chapter 5: Mounting and using filesystems

Once a disk has been partitioned, Linux needs some way to access the data on the partitions. Unlike DOS or Windows (where this is done by assigning a drive letter to each partition), Linux uses a unified directory tree where each partition is mounted at a mount point in that tree. A mount point is a directory that is used as a way to access the filesystem on the partition, and mounting the filesystem is the process of associating a certain filesystem (a partition, for example) with a specific directory in the directory tree. In other words, the first step in managing a storage device is attaching the device to the file system tree. This task can be accomplished on a one-time basis by using tools such as mount (and then unmounted with umount) or persistently across reboots by editing the /etc/fstab file.

The mount command (without any options or arguments) shows the currently mounted filesystems:

```
root@debian:~# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,relatime,size=10240k,nr_inodes=62949,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=51480k,mode=755)
/dev/mapper/debian-root on / type ext4 (rw,relatime,errors=remount-ro,user_xattr,barrier=1,data=ordered)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /run/shm type tmpfs (rw,nosuid,nodev,noexec,relatime,size=102940k)
/dev/sda1 on /boot type ext2 (rw,relatime,errors=continue)
/dev/mapper/debian-home on /home type ext4 (rw,relatime,user_xattr,barrier=1,data=ordered)
/dev/mapper/debian-tmp on /tmp type ext4 (rw,relatime,user_xattr,barrier=1,data=ordered)
/dev/mapper/debian-usr on /usr type ext4 (rw,relatime,user_xattr,barrier=1,data=ordered)
/dev/mapper/debian-var on /var type ext4 (rw,relatime,user_xattr,barrier=1,data=ordered)
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
root@debian:~#
```

In addition, mount is used to mount filesystems into the filesystem tree. Its standard syntax is as follows:

```
mount -t type device dir -o options
```

This command instructs the kernel to mount the filesystem found on device (a partition, for example, that has been formatted with a filesystem type) at the directory dir, using all options. In this form, mount does not look in /etc/fstab for instructions.

If only a directory or device is specified, for example:

```
mount /dir -o options
```

or

```
mount device -o options
```

mount tries to find a mount point and if it can't find any, then searches for a device (both cases in the /etc/fstab file), and finally attempts to complete the mount operation (which usually succeeds, except for the case when either the directory or the device is already being used, or when the user invoking mount is not root).

You will notice that every line in the output of mount has the following format:

```
device on directory type (options)
```

For example,

```
/dev/mapper/debian-home on /home type ext4
(rw,relatime,user_xattr,barrier=1,data=ordered)
```

Reads:

/dev/mapper/debian-home is mounted on /home, which has been formatted as ext4, with the following options: rw,relatime,user_xattr,barrier=1,data=ordered

Mount options

Most frequently used mount options include:

- **async**: allows asynchronous I/O operations on the file system being mounted.
- **auto**: marks the file system as enabled to be mounted automatically using mount -a. It is the opposite of noauto.
- **defaults**: this option is an alias for async,auto,dev,exec,nouser,rw,suid. Note that multiple options must be separated by a comma without any spaces. If by accident you type a space between options, mount will interpret the subsequent text string as another argument.
- **loop**: Mounts an image (an .iso file, for example) as a loop device. This option can be used to simulate the presence of the disk's contents in an optical media reader.
- **noexec**: prevents the execution of executable files on the particular filesystem. It is the opposite of exec.
- **nouser**: prevents any users (other than root) to mount and unmount the filesystem. It is the opposite of user.
- **remount**: mounts the filesystem again in case it is already mounted.
- **ro**: mounts the filesystem as read only.
- **rw**: mounts the file system with read and write capabilities.
- **relatime**: makes access time to files be updated only if atime is earlier than mtime.
- user_xattr: allow users to set and remote extended filesystem attributes.

For example, in order to mount a device with **ro** and **noexec** options, you will need to do:

```
mount -t ext4 /dev/sdg1 /mnt -o ro,noexec
```

In this case we can see that attempts to write a file to or to run a binary file located inside our mounting point fail with corresponding error messages:

```
touch /mnt/myfile
/mnt/bin/echo “Hi there”
```

```
root@debian:~# mount -t ext4 /dev/sdg1 /mnt -o ro,noexec
root@debian:~# touch /mnt/myfile
touch: cannot touch `/mnt/myfile': Read-only file system
root@debian:~# /mnt/bin/echo "Hi there"
bash: /mnt/bin/echo: Permission denied
```

To mount a device with default options:

```
mount -t ext4 /dev/sdg1 /mnt -o defaults
```

In the following scenario, we will try to write a file to our newly mounted device and run an executable file located within its filesystem tree using the same commands as in the previous example:

```
root@debian:~# mount -t ext4 /dev/sdg1 /mnt -o defaults
root@debian:~# touch /mnt/myfile
root@debian:~# ls -l /mnt/myfile
-rw-r--r-- 1 root root 0 Oct 21 20:24 /mnt/myfile
root@debian:~# /mnt/bin/echo "Hi there"
Hi there
root@debian:~#
```

In this last case, it works perfectly.

Unmounting devices

Unmounting a device (with the umount command) means finish writing all the remaining "on transit" data so that it can be safely removed. Note that if you try to remove a mounted device without properly unmounting it first, you run the risk of damaging the device itself or cause data loss.

That being said, in order to unmount a device, you must be "standing outside" its block device descriptor or mount point. In other words, your current working directory must be something else other than the mounting point. Otherwise, you will get a message saying that the device is busy:

```
root@debian:~# cd /mnt
root@debian:/mnt# umount /mnt
umount: /mnt: device is busy.
        (In some cases useful info about processes that use
         the device is found by lsof(8) or fuser(1))
root@debian:/mnt# cd
root@debian:~# umount /mnt
```

An easy way to "leave" the mounting point is typing the cd command which, in lack of arguments, will take us to our current user's home directory, as shown above.

Mounting networked filesystems

The two most frequently used network file systems are SMB (which stands for "Server Message Block") and NFS ("Network File System"). Chances are you will use NFS if you need to set up a share for Unix-like clients

only, and will opt for Samba if you need to share files with Windows-based clients and perhaps other Unix-like clients as well.

The following steps assume that Samba and NFS shares have already been set up in the server with IP 192.168.0.10 (please note that setting up a NFS share is one of the competencies required for the LFCE exam, which we will cover after the present book).

To mount a Samba share on Linux, follow these steps:

STEP 1: Install the samba-client samba-common and cifs-utils packages

Red Hat-based distributions:

```
yum update && yum install samba-client samba-common cifs-utils
```

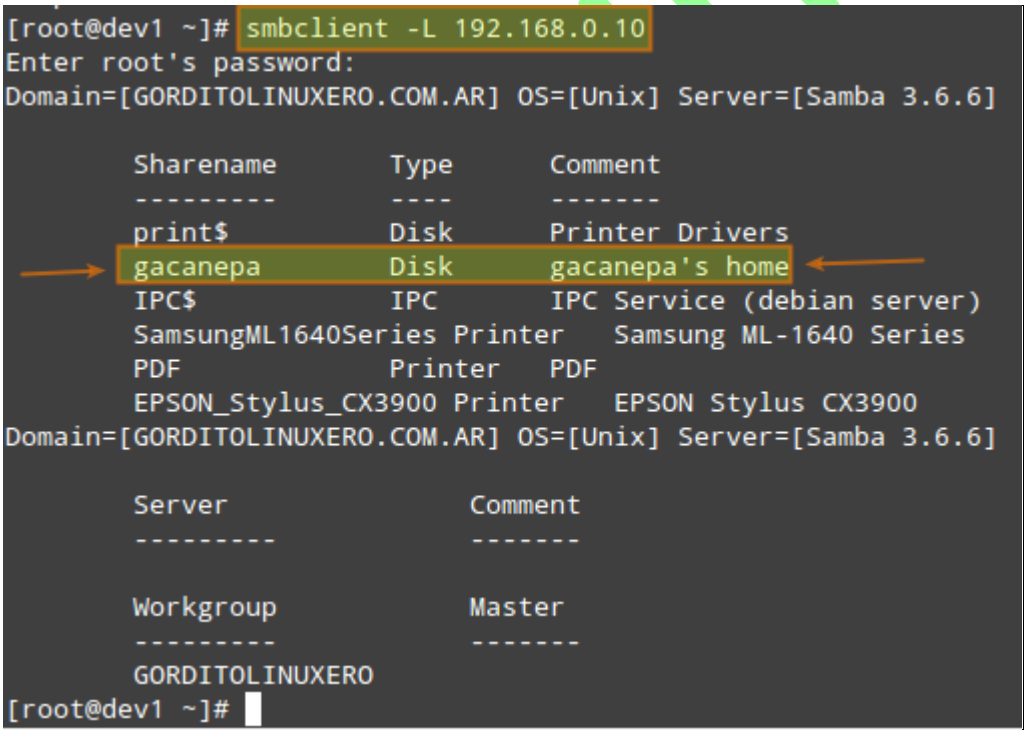
Debian and derivatives:

```
aptitude update && aptitude install samba-client samba-common cifs-utils
```

Then run the following command to look for available samba shares in the server:

```
smbclient -L 192.168.0.10
```

and enter the password for the root account in the remote machine:



In the above image we have highlighted the share that is ready for mounting on our local system. You will need a valid samba username and password on the remote server in order to access it.

STEP 2: When mounting a password-protected network share, it is not a good idea to write your credentials in the /etc/fstab file. Instead, you can store them in a hidden file somewhere with permissions set to 600, like so:

```
mkdir /media/samba
echo “username=samba_username” > /media/samba/.smbcredentials
echo “password=samba_password” >> /media/samba/.smbcredentials
chmod 600 /media/samba/.smbcredentials
```

STEP 3: Then add the following line to /etc/fstab

```
//192.168.0.10/gacanepa /media/samba cifs
credentials=/media/samba/.smbcredentials,defaults 0 0
```

STEP 4: You can now mount your samba share, either manually (mount //192.168.0.10/gacanepa) or by rebooting your machine so as to apply the changes made in /etc/fstab permanently.

```
[root@dev1 ~]# mount | grep gacanepa
[root@dev1 ~]# mount //192.168.0.10/gacanepa
[root@dev1 ~]# mount | grep gacanepa
//192.168.0.10/gacanepa on /media/samba type cifs (rw,relatime,vers=1
0,unix,posixpaths,serverino,acl,rsize=1048576,wsize=65536,actimeo=1)
[root@dev1 ~]#
```

To mount a NFS share, do:

STEP 1: Install the nfs-common and portmap packages

Red Hat-based distributions:

```
yum update && yum install nfs-utils nfs-utils-lib
```

Debian and derivatives:

```
aptitude update && aptitude install nfs-common
```

STEP 2: Create a mounting point for the NFS share

```
mkdir /media/nfs
```

STEP 3: Add the following line to /etc/fstab

```
192.168.0.10:/NFS-SHARE /media/nfs nfs defaults 0 0
```

STEP 4: You can now mount your NFS share, either manually (mount 192.168.0.10:/NFS-SHARE) or by rebooting your machine so as to apply the changes made in /etc/fstab permanently.

```
[root@dev1 ~]# mount | grep NFS
[root@dev1 ~]# mount 192.168.0.10:/NFS-SHARE
[root@dev1 ~]# mount | grep NFS
192.168.0.10:/NFS-SHARE on /media/nfs type nfs4 (rw,relatime
8,wsize=32768,namlen=255,hard,proto=tcp,port=0,timeo=600,ret
taddr=192.168.0.17,local_lock=none,addr=192.168.0.10)
[root@dev1 ~]#
```

Mounting filesystems persistently

As shown in the previous two examples, the /etc/fstab file controls how Linux provides access to disk partitions and removable media devices and consists of a book of lines that contain six fields each; the fields are separated by one or more spaces or tabs. A line that begins with a hash mark (#) is a comment and is ignored.

Each line has the following format:

```
<file system> <mount point> <type> <options> <dump> <pass>
```

where:

<filesystem>: The first column specifies the mount device. Most distributions now specify partitions by their labels or UUIDs. This practice can help reduce problems if partition numbers change.

<mount point>: The second column specifies the mount point.

<type>: The file system type code is the same as the type code used to mount a filesystem with the mount command. A file system type code of auto lets the kernel auto-detect the filesystem type, which can be a convenient option for removable media devices. Note that this option may not be available for all filesystems out there.

<options>: One (or more) mount option(s).

<dump>: You will most likely leave this to 0 (otherwise set it to 1) to disable the dump utility to backup the filesystem upon boot (The dump program was once a common backup tool, but it is much less popular today.)

<pass>: This column specifies whether the integrity of the filesystem should be checked at boot time with fsck. A 0 means that fsck should not check a filesystem. The higher the number, the lowest the priority. Thus, the root partition will most likely have a value of 1, while all others that should be checked should have a value of 2.

Examples

To mount a partition with label TECMINT at boot time with rw and noexec attributes, you should add the following line in /etc/fstab:

```
LABEL=TECMINT /mnt ext4 rw,noexec 0 0
```

If you want the contents of a disk in your DVD drive be available at boot time:

```
/dev/sr0    /media/cdrom0    iso9660    ro,user,noauto    0    0
```

where /dev/sr0 is your DVD drive.

Summary

You can rest assured that mounting and unmounting local and network filesystems from the command line will be part of your day-to-day responsibilities as sysadmin. You will also need to master /etc/fstab. For more information on this essential system file, you may want to check the Arch Linux documentation on the subject at https://wiki.archlinux.org/index.php/fstab.

Chapter 6: RAIDs and backups

The technology known as Redundant Array of Independent Disks (RAID) is a storage solution that combines multiple hard disks into a single logical unit to provide redundancy of data and/or improve performance in read / write operations to disk.

However, the actual fault-tolerance and disk I/O performance lean on how the hard disks are set up to form the disk array. Depending on the available devices and the fault tolerance / performance needs, different RAID levels are defined.

Our tool of choice for creating, assembling, managing, and monitoring our software RAIDs is called **mdadm** (short for multiple disks admin).

```
aptitude update && aptitude install mdadm # Debian and derivatives
yum update && yum install mdadm # CentOS
zypper refresh && zypper install mdadm # openSUSE
```

Assembling partitions as RAID devices

The process of assembling existing partitions as RAID devices consists of the following steps:

1) Create the array using **mdadm**. If one of the partitions has been formatted previously, or has been a part of another RAID array previously, you will be prompted to confirm the creation of the new array. Assuming you have taken the necessary precautions to avoid losing important data that may have resided in them, you can safely type **y** and press Enter.

```
root@dev2:~# mdadm --create --verbose /dev/md0 --level=stripe --raid-devices=2 /dev/sdb1 /dev/sd
mdadm: chunk size defaults to 512K
mdadm: /dev/sdb1 appears to contain an ext2fs file system
    size=8387564K  mtime=Thu Oct 23 10:05:03 2014
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
root@dev2:~#
```

2) In order to check the array creation status, you will use the following commands – regardless of the RAID type. These are just as valid as when we are creating a RAID0 (as shown above), or when you are in the process of setting up a RAID5, as shown in the image below.

```
cat /proc/mdstat
```

or more detailed with

```
mdadm --detail /dev/md0
```

```
root@dev2:~# cat /proc/mdstat
Personalities : [raid0] [raid1] [raid6] [raid5] [raid4]
md0 : active raid5 sdd1[4] sde1[3](S) sdc1[1] sdb1[0]
      16765952 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>
root@dev2:~# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Thu Oct 23 22:03:41 2014
     Raid Level : raid5
     Array Size : 16765952 (15.99 GiB 17.17 GB)
  Used Dev Size : 8382976 (7.99 GiB 8.58 GB)
   Raid Devices : 3
  Total Devices : 4
    Persistence : Superblock is persistent

    Update Time : Thu Oct 23 22:12:43 2014
          State : clean
 Active Devices : 3
Working Devices : 4
 Failed Devices : 0
  Spare Devices : 1

         Layout : left-symmetric
     Chunk Size : 512K

           Name : dev2:0  (local to host dev2)
           UUID : 03b6375b:92b3f863:d43083fe:a40913b8
         Events : 24

    Number   Major   Minor   RaidDevice State
       0       8       17        0      active sync   /dev/sdb1
       1       8       33        1      active sync   /dev/sdc1
       4       8       49        2      active sync   /dev/sdd1

       3       8       65        -      spare   /dev/sde1
root@dev2:~#
```

3) Format the device with a filesystem as per your needs / requirements, as explained in Chapter 4 (“Partitioning storage devices, formatting filesystems, and configuring swap partitions”) of this book.

4) Instruct the monitoring service to “keep an eye” on the array. Add the output of

```
mdadm --detail --scan
```

to /etc/mdadm/mdadm.conf (Debian and derivatives) or /etc/mdadm.conf (CentOS / openSUSE), like so:

```
root@dev2:~# mdadm --detail --scan
ARRAY /dev/md0 metadata=1.2 spares=1 name=dev2:0 UUID=190be429:40d8d371:cf2272e9:6253cb96
root@dev2:~#
```

```
# definitions of existing MD arrays
ARRAY /dev/md0 metadata=1.2 spares=1 name=dev2:0 UUID=190be429:40d8d371:cf2272e9:6253cb96
```

```
mdadm --assemble --scan # Assemble the array
```

To ensure the service starts on system boot, run the following commands as root:

Debian and derivatives

```
update-rc.d mdadm defaults # Debian and derivatives, though it should start running on boot by default
```

Edit the /etc/default/mdadm file and add the following line:

```
AUTOSTART=true
```

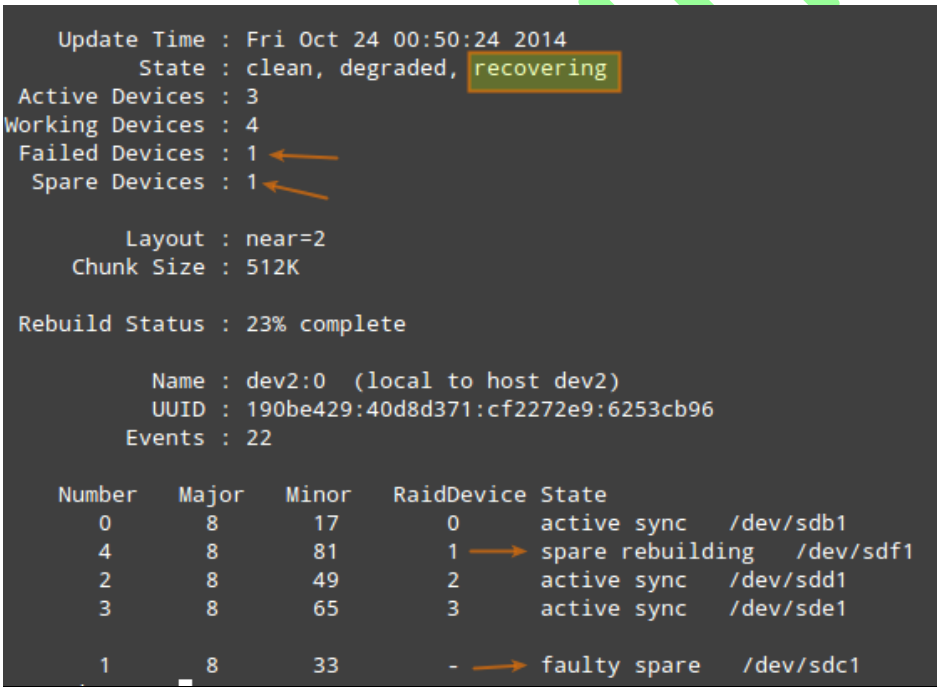
On CentOS and openSUSE (systemd-based)

```
systemctl start mdmonitor
systemctl enable mdmonitor
```

On CentOS and openSUSE (SysVinit-based)

```
service mdmonitor start
chkconfig mdmonitor on
```

5) In RAID levels that support redundancy, replace failed drives when needed. When a device in the disk array becomes faulty, a rebuild automatically starts only if there was a spare device added when we first created the array:



Otherwise, we need to manually attach an extra physical drive to our system and run

```
mdadm /dev/md0 --add /dev/sdX1
```

where /dev/md0 is the array that experienced the issue and /dev/sdX1 is the new device.

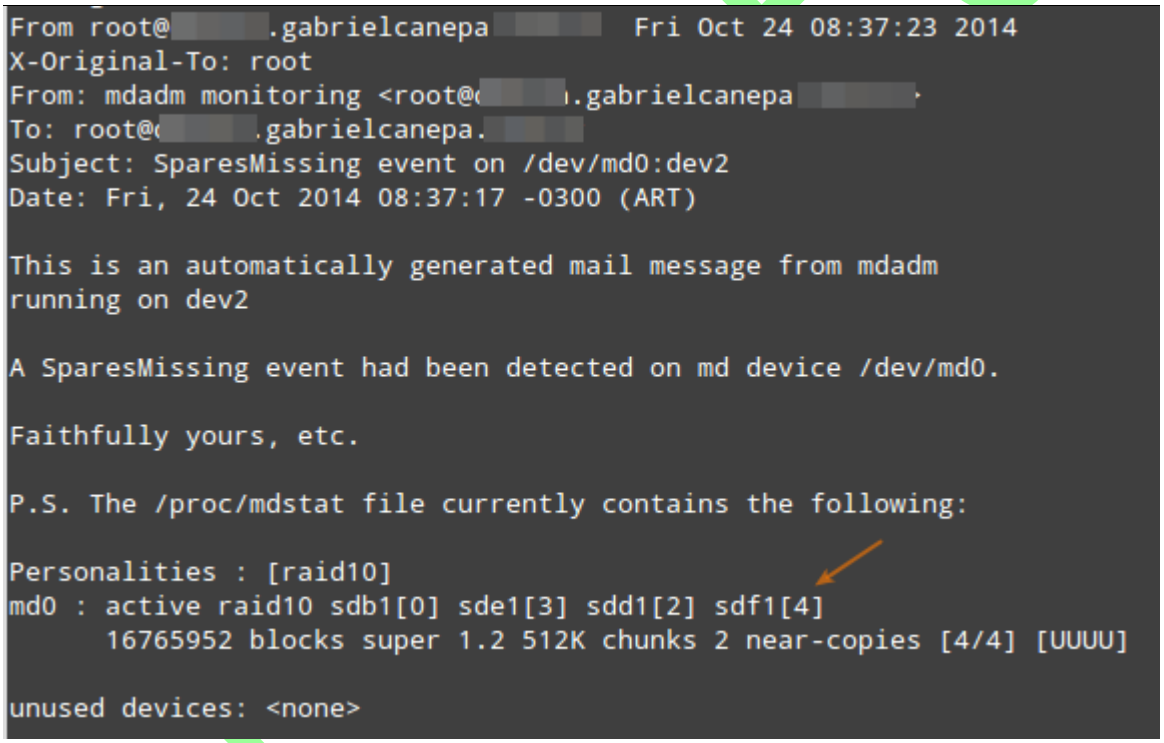
6) [OPTIONAL] Disassemble a working array. You may have to do this if you need to create a new array using the devices

```
mdadm --stop /dev/md0 # Stop the array
mdadm --remove /dev/md0 # Remove the RAID device
mdadm --zero-superblock /dev/sdX1 # Overwrite the existing md superblock with zeroes
```

7) [OPTIONAL] Set up mail alerts. You can configure a valid email address or system account to send alerts to (make sure you have this line in mdadm.conf):

```
MAILADDR root
```

In this case, all alerts that the RAID monitoring daemon collects will be sent to the local root account's mail box. One of such alerts looks like the following (please note that this event is related to the example in STEP 5, where a device was marked as faulty and the spare device was automatically built into the array by **mdadm**. Thus, we "ran out" of healthy spare devices and we got the alert):



RAID Levels

- RAID 0 (aka stripe): the total array size is n times the size of the smallest partition, where n is the number of independent disks in the array (you will need at least two drives). Run the following command to assemble a RAID 0 array using partitions /dev/sdb1 and /dev/sdc1:

```
mdadm --create --verbose /dev/md0 --level=stripe --raid-devices=2 /dev/sdb1
/dev/sdc1
```

Common uses: Setups that support real-time applications where performance is more important than fault-tolerance.

- RAID 1 (aka mirror): the total array size equals the size of the smallest partition (you will need at least two drives). Run the following command to assemble a RAID 1 array using partitions /dev/sdb1 and /dev/sdc1:

```
mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
```

Common uses: Installation of the operating system or important subdirectories, such as /home.

- RAID 5 (aka drives with parity): the total array size will be (n - 1) times the size of the smallest partition. The "lost" space in (n-1) is used for parity (redundancy) calculation (you will need at least three drives). Note that you can specify a spare device (/dev/sde1 in this case) to replace a faulty part when an issue occurs. Run the following command to assemble a RAID 5 array using partitions /dev/sdb1, /dev/sdc1, /dev/sdd1, and /dev/sde1 as spare:

```
mdadm --create --verbose /dev/md0 --level=5 --raid-devices=3 /dev/sdb1 /dev/sdc1
/dev/sdd1 --spare-devices=1 /dev/sde1
```

Common uses: Web and file servers.

- RAID 6 (aka drives with double parity): the total array size will be (n*s)-2*s, where n is the number of independent disks in the array and s is the size of the smallest disk. Note that you can specify a spare device (/dev/sdf1 in this case) to replace a faulty part when an issue occurs. Run the following command to assemble a RAID 6 array using partitions /dev/sdb1, /dev/sdc1, /dev/sdd1, /dev/sde1, and /dev/sdf1 as spare:

```
mdadm --create --verbose /dev/md0 --level=6 --raid-devices=4 /dev/sdb1 /dev/sdc1
/dev/sdd1 /dev/sde --spare-devices=1 /dev/sdf1
```

Common uses: File and backup servers with large capacity and high availability requirements.

- RAID 1+0 (aka stripe of mirrors): the total array size is computed based on the formulas for RAID 0 and RAID 1, since RAID 1+0 is a combination of both. First, calculate the size of each mirror and then the size of the stripe. Note that you can specify a spare device (/dev/sdf1 in this case) to replace a faulty part when an issue occurs. Run the following command to assemble a RAID 1+0 array using partitions /dev/sdb1, /dev/sdc1, /dev/sdd1, /dev/sde1, and /dev/sdf1 as spare:

```
mdadm --create --verbose /dev/md0 --level=10 --raid-devices=4 /dev/sd[b-e]1 --
spare-devices=1 /dev/sdf1
```

Common uses: Database and application servers that require fast I/O operations.

Creating and managing system backups

It never hurts to remember that RAID with all its bounties IS NOT A REPLACEMENT FOR BACKUPS! Write it 1000 times on the chalkboard if you need to, but make sure you keep that idea in mind at all times.

Before we begin, we must note that there is no one-size-fits-all solution for system backups, but here are some things that you do need to take into account while planning a backup strategy:

1) What do you use your system for? (Desktop or server? If the latter case applies, what are the most critical services - whose configuration would be a real pain to lose?)

2) How often do you need to take backups of your system?

3) What is the data (e.g. files / directories / database dumps) that you want to backup? You may also want to consider if you really need to backup huge files (such as audio or video files).

4) Where (meaning physical place and media) will those backups be stored?

Backing up your data

Method 1: Backup entire drives with dd. You can either back up an entire hard disk or a partition by creating an exact image at any point in time. Note that this works best when the device is offline, meaning it's not mounted and there are no processes accessing it for I/O operations. The downside of this backup approach is that the image will have the same size as the disk or partition, even when the actual data occupies a small percentage of it. For example, if you want to image a partition of 20 GB that is only 10% full, the image file will still be 20 GB in size. In other words, it's not only the actual data that gets backed up, but the entire partition itself. You may consider using this method if you need exact backups of your devices.

Creating an image file out of an existing device

```
dd if=/dev/sda of=/system_images/sda.img
```

or

```
dd if=/dev/sda | gzip -c > /system_images/sda.img.gz # Alternatively, you can compress the image file
```

Restoring the backup from the image file

```
dd if=/system_images/sda.img of=/dev/sda
```

or (depending on your choice while creating the image)

```
gzip -dc /system_images/sda.img.gz | dd of=/dev/sda
```

Method 2: Backup certain files / directories with tar - already covered in Part 3 of this series. You may consider using this method if you need to keep copies of specific files and directories (configuration files, users' home directories, and so on).

Method 3: Synchronize files with rsync. Rsync is a versatile remote (and local) file-copying tool. If you need to backup and synchronize your files to/from network drives, rsync is a go.

Whether you're synchronizing two local directories or local < --- > remote directories mounted on the local filesystem, the basic syntax is the same.

Synchronizing two local directories or local < --- > remote directories mounted on the local filesystem

```
rsync -av source_directory destination directory
```

where

-a: recurse into subdirectories (if they exist), preserve symbolic links, timestamps, permissions, and original owner / group.

-v: verbose

```
root@dev2:~# mkdir /home/gacanepa/test
root@dev2:~# rsync -av /home/gacanepa backups
sending incremental file list
gacanepa/
gacanepa/test/

sent 305 bytes  received 21 bytes  652.00 bytes/sec
total size is 22519  speedup is 69.08
root@dev2:~# ls -a backups/gacanepa
.   .aptitude      .bash_logout  .lesshst  .profile  .viminfo
..  .bash_history  .bashrc       mbox      test
root@dev2:~#
```

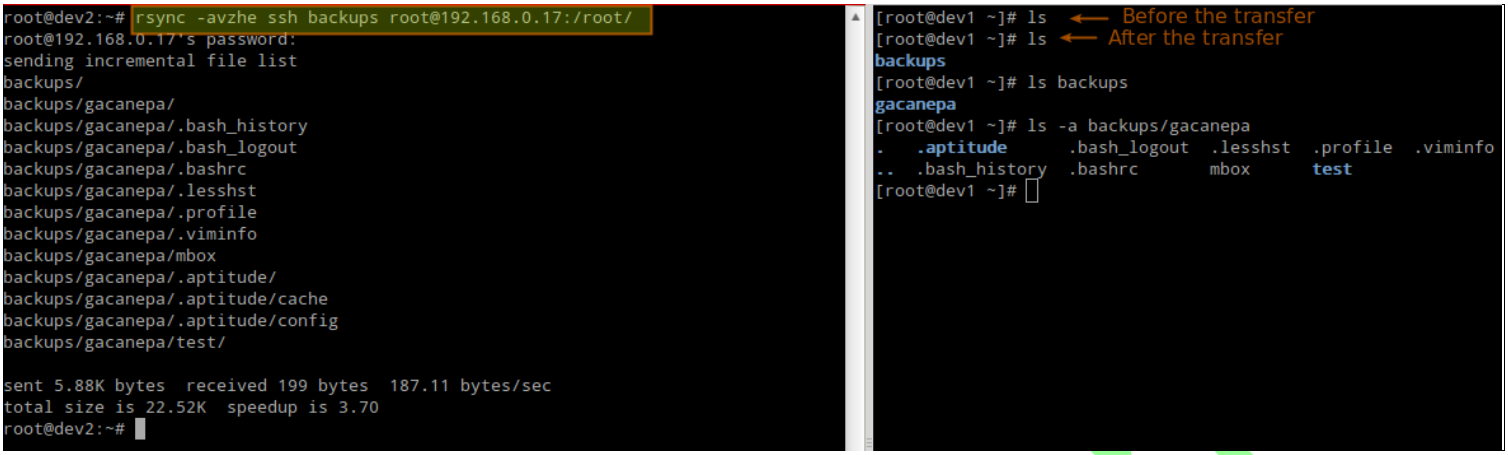
In addition, if you want to increase the security of the data transfer over the wire, you can use ssh over rsync.

To synchronize local → remote directories over ssh, do:

```
rsync -avzhe ssh backups root@remote_host:/remote_directory/
```

This example will synchronize the **backups** directory on the local host with the contents of **/root/remote_directory** on the remote host.

where the -h option shows file sizes in human-readable format, and the -e flag is used to indicate a ssh connection.



To synchronize remote → local directories (switch the source and destination directories from the previous example) over ssh, do:

```
rsync -avzhe ssh root@remote_host:/remote_directory/ backups
```

Please note that these are only 3 examples (most frequent cases you're likely to run into) of the use of rsync.

Summary

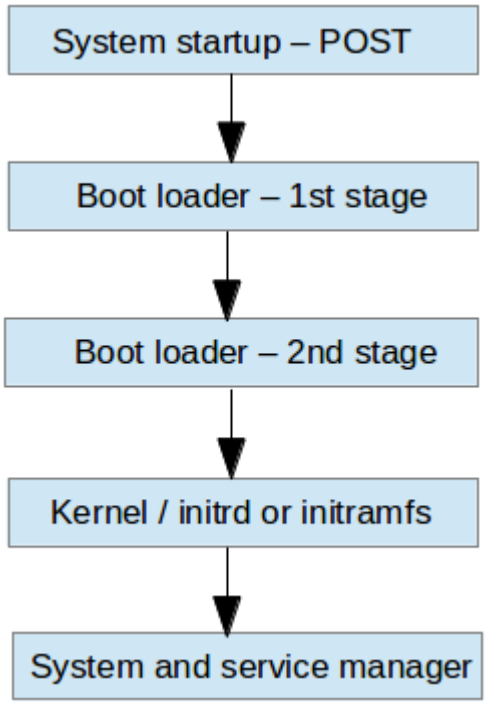
As a sysadmin, you need to ensure that your systems perform as good as possible. If you're well prepared, and if the integrity of your data is well supported by a storage technology such as RAID and regular system backups, you'll be safe.

Chapter 7: Managing the startup process and related services

The boot process of a Linux system consists of several phases, each represented by a different component. In this chapter we will share an overall description of each one of them.

The boot process

The following diagram briefly summarizes the boot process and shows all the main components involved:



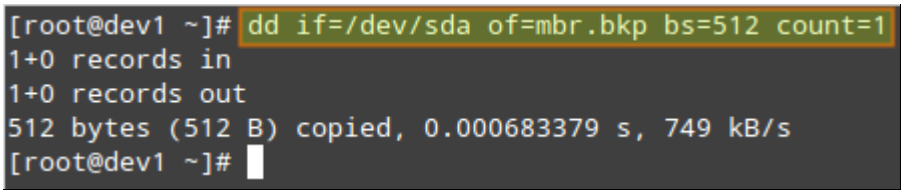
When you press the Power button on your machine, the firmware that is stored in a EEPROM chip in the motherboard initializes the POST (Power-On Self Test) to check on the state of the system's hardware resources. When the POST is finished, the firmware then searches and loads the 1st stage boot loader, located in the MBR or in the EFI partition of the first available disk, and gives control to it.

1) The MBR is located in the first sector of the disk marked as bootable in the BIOS settings and is 512 bytes in size:

- First 446 bytes: The bootloader contains both executable code and error message text.
- Next 64 bytes: The Partition table contains a record for each of four partitions (primary or extended). Among other things, each record indicates the status (active / not active), size, and start / end sectors of each partition.
- Last 2 bytes: The magic number serves as a validation check of the MBR.

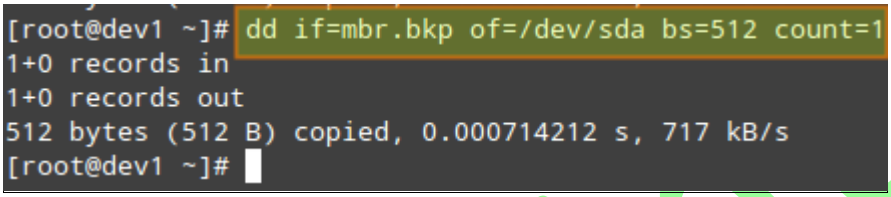
The following command performs a backup of the MBR (in this example, /dev/sda is the first hard disk). The resulting file, mbr.bkp can come in handy should the partition table become corrupt, for example, rendering the system unbootable. Of course, in order to use it later if the need arises, we will need to save it and store it somewhere else (like a USB drive, for example). That file will help us restore the MBR and will get us going once again if and only if we do not change the hard drive layout in the meanwhile.

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



and restore it with

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



2) For systems using the EFI/UEFI method, the UEFI firmware reads its settings to determine which UEFI application is to be launched and from where (i.e., in which disk and partition the EFI partition is located).

Next, the 2nd stage boot loader (aka boot manager) is loaded and run. GRUB [GRand Unified Boot] is the most frequently used boot manager in Linux. One of two distinct versions can be found on most systems used today:

1. GRUB legacy configuration file: /boot/grub/menu.lst (older distributions, not supported by EFI/UEFI firmwares)
2. GRUB2 configuration file: most likely, /etc/default/grub.

Although the objectives of the LFCS exam do not explicitly request knowledge about GRUB internals, if you're brave and can afford to mess up your system (you may want to try it first on a virtual machine, just in case), you need to run

```
update-grub
```

as root after modifying GRUB's configuration in order to apply the changes.

Basically, GRUB loads the default kernel and the initrd or initramfs image. In few words, initrd or initramfs help to perform the hardware detection, the kernel module loading and the device discovery necessary to get the real root filesystem mounted.

Once the real root filesystem is up, the kernel executes the system and service manager (init or systemd, whose process identification or PID is always 1) to begin the normal user-space boot process in order to present a user

interface. Both init and systemd are daemons (background processes) that manage other daemons, as the first service to start (during boot) and the last service to terminate (during shutdown).

Starting services

The concept of runlevels in Linux specifies different ways to use a system by controlling which services are running. In other words, a runlevel controls what tasks can be accomplished in the current execution state = runlevel (and which ones cannot).

Traditionally, this startup process was performed based on conventions that originated with System V UNIX, with the system passing executing collections of scripts that start and stop services as the machine entered a specific runlevel (which, in other words, is a different mode of running the system).

Within each runlevel, individual services can be set to run, or to be shut down if running. Latest versions of some major distributions are moving away from the System V standard in favor of a rather new service and system manager called systemd (which stands for system daemon), but usually support sysv commands for compatibility purposes. This means that you can run most of the well-known sysv init tools in a systemd-based distribution.

Besides starting the system process, init looks to the /etc/inittab file to decide what runlevel must be entered:

Runlevel	Description
0	Halt the system. Runlevel 0 is a special transitional state used to shutdown the system quickly.
1	Also aliased to s, or S, this runlevel is sometimes called maintenance mode. What services, if any, are started at this runlevel varies by distribution. It's typically used for low-level system maintenance that may be impaired by normal system operation.
2	Multiuser. On Debian systems and derivatives, this is the default runlevel, and includes -if available- a graphical login. On Red-Hat based systems, this is multiuser mode without networking.
3	On Red-Hat based systems, this is the default multiuser mode, which runs everything except the graphical environment. This runlevel and levels 4 and 5 usually are not used on Debian-based systems.
4	Typically unused by default and therefore available for customization.
5	On Red-Hat based systems, full multiuser mode with GUI login. This runlevel is like level 3, but with a GUI login available.
6	Reboot the system.

To switch between runlevels, we can simply issue a runlevel change using the init command: init N (where N is one of the runlevels listed above). Please note that this is not the recommended way of taking a running system to a different runlevel because it gives no warning to existing logged-in users (thus causing them to lose work and processes to terminate abnormally). Instead, the shutdown command should be used to restart the system (which first sends a warning message to all logged-in users and blocks any further logins; it then signals init to

switch runlevels); however, the default runlevel (the one the system will boot to) must be edited in the /etc/inittab file first.

For that reason, follow these steps to properly switch between runlevels:

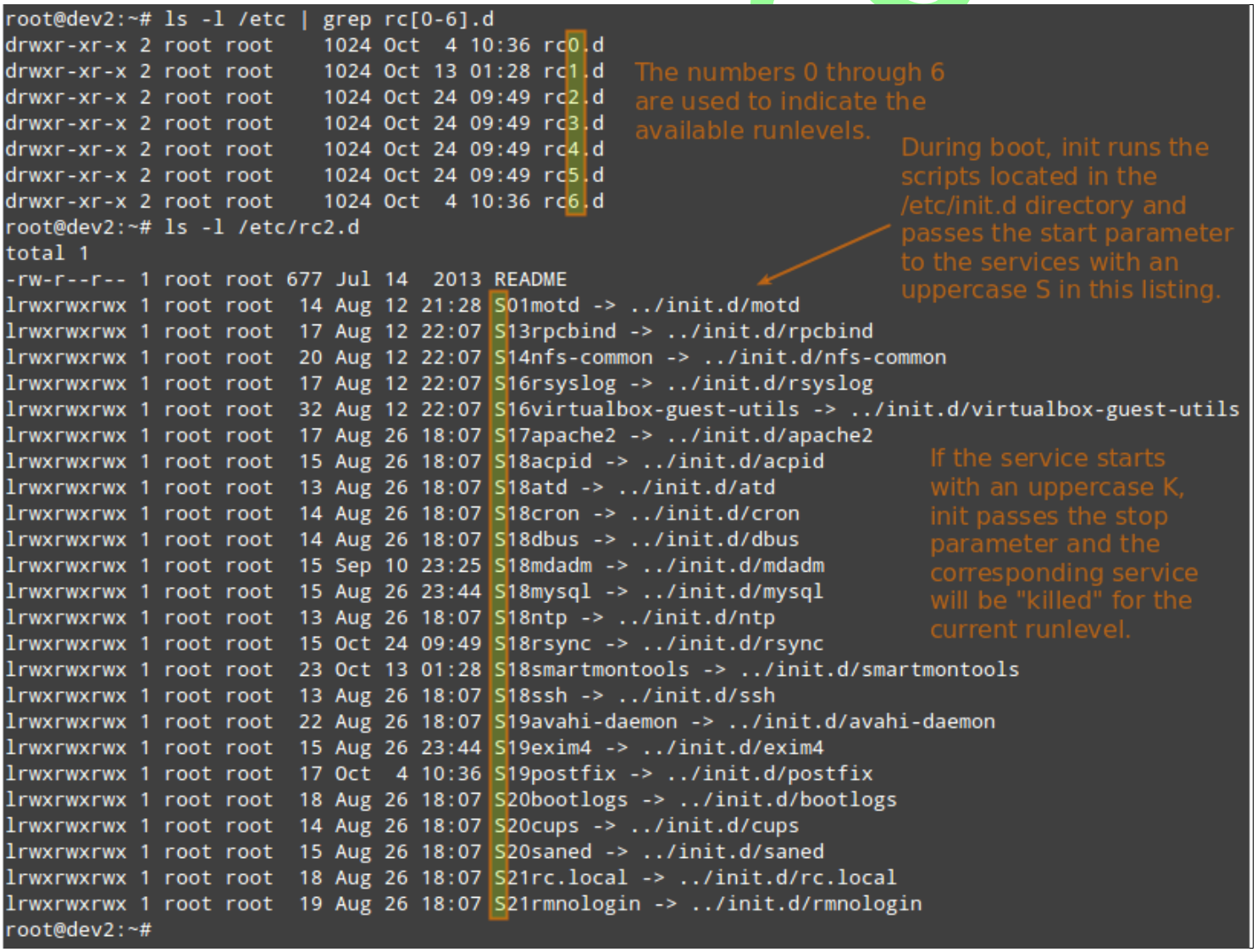
As root, look for the following line in /etc/inittab

```
id:2:initdefault:
```

and change the number 2 for the desired runlevel with your preferred text editor, such as vim. Next, run as root:

```
shutdown -r now
```

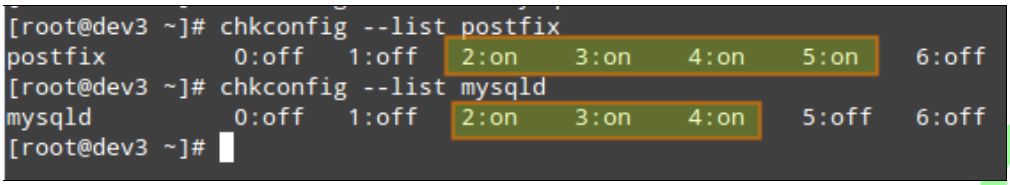
That last command will restart the system, causing it to start in the specified runlevel during next boot, and will run the scripts located in the /etc/rc[runlevel].d directory in order to decide which services should be started and which ones should not. For example, for runlevel 2 in the following system:



To enable or disable system services on boot, we will use chkconfig in CentOS / openSUSE and sysv-rc-conf in Debian and derivatives. This tool can also show us what is the preconfigured state of a service for a particular runlevel.

In order to list the runlevel configuration for a service, do

```
chkconfig --list [service name]
chkconfig --list postfix
chkconfig --list mysqld
```



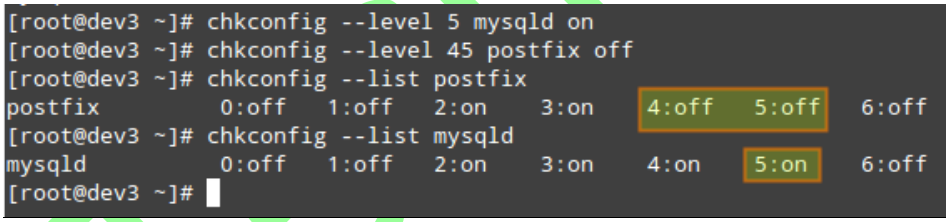
In the above image we can see that postfix is set to start when the system enters runlevels 2 through 5, whereas mysqld will be running by default for runlevels 2 through 4. Now suppose that this is not the expected behavior. For example, we need to turn on mysqld for runlevel 5 as well, and turn off postfix for runlevels 4 and 5. Here's what we would do in each case (run the following commands as root):

Enabling a service for a particular runlevel

```
chkconfig --level [level(s)] service on
chkconfig --level 5 mysqld on
```

Disabling a service for particular runlevels

```
chkconfig --level [level(s)] service off
chkconfig --level 45 postfix off
```



We will now perform similar tasks in a Debian-based system.

Configuring a service to start automatically on a specific runlevel and prevent it from starting on all others

STEP 1: Let's use the following command to see what are the runlevels where mdadm is configured to start:

```
ls -l /etc/rc[0-6].d | grep -E 'rc[0-6]|mdadm'
```

```
root@dev2:~# ls -l /etc/rc[0-6].d | grep -E 'rc[0-6]|mdadm'
/etc/rc0.d: Runlevel 0
lrwxrwxrwx 1 root root  15 Oct 25 20:14 K01mdadm -> ../init.d/mdadm
lrwxrwxrwx 1 root root  20 Sep 10 23:25 K09mdadm-raid -> ../init.d/mdadm-raid
lrwxrwxrwx 1 root root  24 Sep 10 23:25 K10mdadm-waitidle -> ../init.d/mdadm-waitidle
/etc/rc1.d: Runlevel 1
lrwxrwxrwx 1 root root  15 Oct 25 20:14 K01mdadm -> ../init.d/mdadm
/etc/rc2.d: Runlevel 2
lrwxrwxrwx 1 root root  15 Oct 25 20:14 S18mdadm -> ../init.d/mdadm
/etc/rc3.d: Runlevel 3
lrwxrwxrwx 1 root root  15 Oct 25 20:14 S18mdadm -> ../init.d/mdadm
/etc/rc4.d: Runlevel 4
lrwxrwxrwx 1 root root  15 Oct 25 20:14 S18mdadm -> ../init.d/mdadm
/etc/rc5.d: Runlevel 5
lrwxrwxrwx 1 root root  15 Oct 25 20:14 S18mdadm -> ../init.d/mdadm
/etc/rc6.d: Runlevel 6
lrwxrwxrwx 1 root root  15 Oct 25 20:14 K01mdadm -> ../init.d/mdadm
lrwxrwxrwx 1 root root  20 Sep 10 23:25 K09mdadm-raid -> ../init.d/mdadm-raid
lrwxrwxrwx 1 root root  24 Sep 10 23:25 K10mdadm-waitidle -> ../init.d/mdadm-waitidle
root@dev2:~#
```

An uppercase S indicates that the corresponding service will be started when the system enters the corresponding runlevel, whereas a lowercase K means the service will not be started.

STEP 2: We will use sysv-rc-conf to prevent mdadm from starting on all runlevels except 2. Just check or uncheck (with the space bar) as desired (you can move up, down, left, and right with the arrow keys)

```
sysv-rc-conf
```

```
SysV Runlevel Config   -: stop service  =/+: start service  h: help  q: quit

service      1       2       3       4       5       0       6       S
--------------------------------------------------------------------------
kmod         [ ]     [ ]     [ ]     [ ]     [ ]     [ ]     [ ]     [X]
lvm2         [ ]     [ ]     [ ]     [ ]     [ ]     [ ]     [ ]     [X]
mdadm        [ ]     [X]     [ ]     [ ]     [ ]     [ ]     [ ]     [ ]
mdadm-raid   [ ]     [ ]     [ ]     [ ]     [ ]     [ ]     [ ]     [X]
mdadm-wai$   [ ]     [ ]     [ ]     [ ]     [ ]     [ ]     [ ]     [ ]
motd         [X]     [X]     [X]     [X]     [X]     [ ]     [ ]     [ ]
```

then press q to quit.

STEP3: We will restart the system and run again the command from STEP 1.

```
root@dev2:~# ls -l /etc/rc[0-6].d | grep -E 'rc[0-6]|mdadm'
/etc/rc0.d:
lrwxrwxrwx 1 root root  15 Oct 25 21:56 K01mdadm -> ../init.d/mdadm
lrwxrwxrwx 1 root root  20 Sep 10 23:25 K09mdadm-raid -> ../init.d/mdadm-raid
lrwxrwxrwx 1 root root  24 Sep 10 23:25 K10mdadm-waitidle -> ../init.d/mdadm-waitidle
/etc/rc1.d:
lrwxrwxrwx 1 root root  15 Oct 25 21:56 K01mdadm -> ../init.d/mdadm
/etc/rc2.d:
lrwxrwxrwx 1 root root  15 Oct 25 21:56 S18mdadm -> ../init.d/mdadm
/etc/rc3.d:
lrwxrwxrwx 1 root root  15 Oct 25 22:06 K01mdadm -> ../init.d/mdadm
/etc/rc4.d:
lrwxrwxrwx 1 root root  15 Oct 25 22:06 K01mdadm -> ../init.d/mdadm
/etc/rc5.d:
lrwxrwxrwx 1 root root  15 Oct 25 22:06 K01mdadm -> ../init.d/mdadm
/etc/rc6.d:
lrwxrwxrwx 1 root root  15 Oct 25 21:56 K01mdadm -> ../init.d/mdadm
lrwxrwxrwx 1 root root  20 Sep 10 23:25 K09mdadm-raid -> ../init.d/mdadm-raid
lrwxrwxrwx 1 root root  24 Sep 10 23:25 K10mdadm-waitidle -> ../init.d/mdadm-waitidle
root@dev2:~#
```

in the above image we can see that mdadm is configured to start only on runlevel 2.

Important: In Ubuntu 14.04, the /etc/inittab file has been replaced by /etc/init/rc-sysinit.conf. While the involved principles are basically the same, you may want to check the Upstart HowTo in the Ubuntu official help documentation.

What about systemd?

systemd is another service and system manager that is being adopted by several major Linux distributions. It aims to allow more processing to be done in parallel during system startup (unlike sysvinit, which always tends to be slower because it starts processes one at a time, checks whether one depends on another, and waits for daemons to launch so more services can start), and to serve as a dynamic resource management to a running system. Thus, services are started when needed (to avoid consuming system resources) instead of being launched without a solid reason during boot.

Viewing the status of all the processes running on your system, both systemd native and SysV services, run the following command:

```
systemctl
```

```
[root@dev1 ~]# systemctl
UNIT                                                                LOAD   ACTIVE SUB
proc-sys-fs-binfmt_misc.automount                                   loaded active waiting
sys-devices-pci0000:00-...a2-host1-target1:0:0-1:0:0:0-block-sr0.device loaded active plugged
sys-devices-pci0000:00-0000:00:03.0-net-enp0s3.device               loaded active plugged
sys-devices-pci0000:00-0000:00:05.0-sound-card0.device              loaded active plugged
sys-devices-pci0000:00-...st2-target2:0:0-2:0:0:0-block-sda-sda1.device loaded active plugged
sys-devices-pci0000:00-...st2-target2:0:0-2:0:0:0-block-sda-sda2.device loaded active plugged
sys-devices-pci0000:00-...a3-host2-target2:0:0-2:0:0:0-block-sda.device loaded active plugged
sys-devices-platform-serial8250-tty-ttyS0.device                    loaded active plugged
sys-devices-platform-serial8250-tty-ttyS1.device                    loaded active plugged
sys-devices-platform-serial8250-tty-ttyS2.device                    loaded active plugged
sys-devices-platform-serial8250-tty-ttyS3.device                    loaded active plugged
sys-devices-virtual-block-dm\x2d0.device                            loaded active plugged
sys-devices-virtual-block-dm\x2d1.device                            loaded active plugged
sys-module-configfs.device                                          loaded active plugged
sys-subsystem-net-devices-enp0s3.device                             loaded active plugged
-.mount                                                             loaded active mounted
boot.mount                                                          loaded active mounted
dev-hugepages.mount                                                 loaded active mounted
dev-mqueue.mount                                                    loaded active mounted
media-nfs.mount                                                     loaded active mounted
media-samba.mount                                                   loaded failed failed
proc-fs-nfsd.mount                                                  loaded active mounted
sys-kernel-config.mount                                             loaded active mounted
sys-kernel-debug.mount                                              loaded active mounted
```

The LOAD column shows whether the unit definition (refer to the UNIT column, which shows the service or anything maintained by systemd) was properly loaded, while the ACTIVE and SUB columns show the current status of such unit.

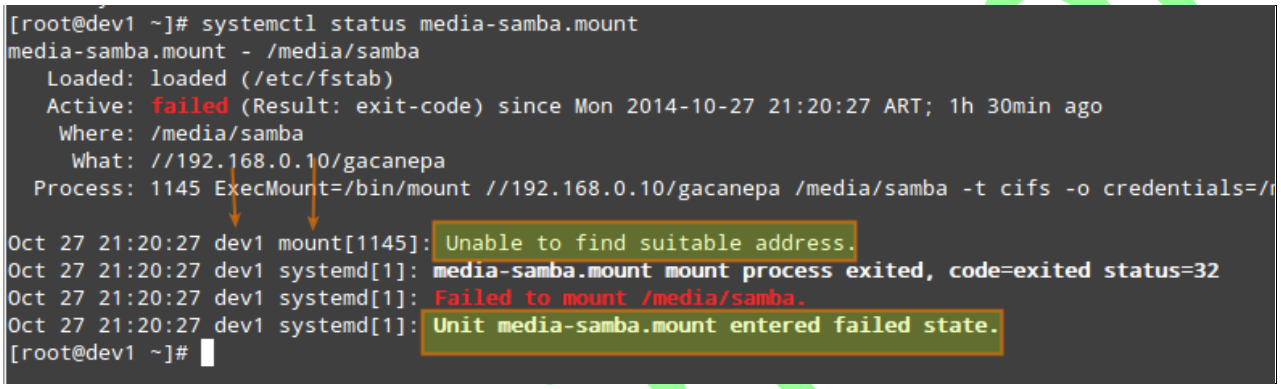
Displaying information about the current status of a service

When the ACTIVE column indicates that an unit's status is other than active, we can check what happened using

```
system status [unit]
```

For example, in the image above, media-samba.mount is in failed state. Let's run

```
systemctl status media-samba.mount
```

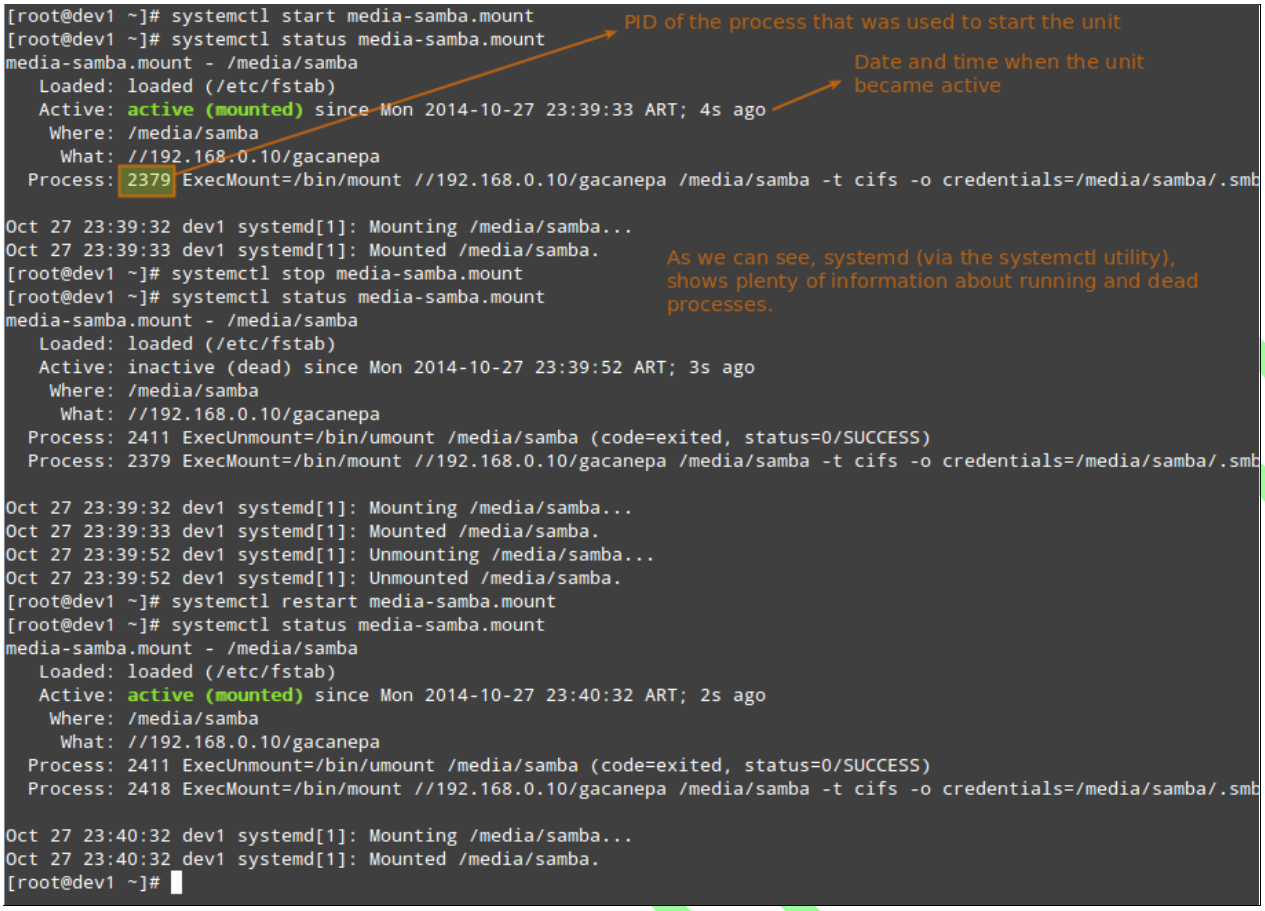


We can see that media-samba.mount failed because the mount process on host dev1 was unable to find the network share at //192.168.0.10/gacanepa.

Starting or stopping services

Once the network share //192.168.0.10/gacanepa becomes available, let's try to start, then stop, and finally restart the unit media-samba.mount. After performing each action, let's run systemctl status media-samba.mount to check on its status:

```
systemctl start media-samba.mount
systemctl status media-samba.mount
systemctl stop media-samba.mount
systemctl restart media-samba.mount
systemctl status media-samba.mount
```




```
[root@dev1 ~]# systemctl start media-samba.mount
[root@dev1 ~]# systemctl status media-samba.mount
media-samba.mount - /media/samba
   Loaded: loaded (/etc/fstab)
   Active: active (mounted) since Mon 2014-10-27 23:39:33 ART; 4s ago
    Where: /media/samba
     What: //192.168.0.10/gacanepa
  Process: 2379 ExecMount=/bin/mount //192.168.0.10/gacanepa /media/samba -t cifs -o credentials=/media/samba/.smb

Oct 27 23:39:32 dev1 systemd[1]: Mounting /media/samba...
Oct 27 23:39:33 dev1 systemd[1]: Mounted /media/samba.
[root@dev1 ~]# systemctl stop media-samba.mount
[root@dev1 ~]# systemctl status media-samba.mount
media-samba.mount - /media/samba
   Loaded: loaded (/etc/fstab)
   Active: inactive (dead) since Mon 2014-10-27 23:39:52 ART; 3s ago
    Where: /media/samba
     What: //192.168.0.10/gacanepa
  Process: 2411 ExecUnmount=/bin/umount /media/samba (code=exited, status=0/SUCCESS)
  Process: 2379 ExecMount=/bin/mount //192.168.0.10/gacanepa /media/samba -t cifs -o credentials=/media/samba/.smb

Oct 27 23:39:32 dev1 systemd[1]: Mounting /media/samba...
Oct 27 23:39:33 dev1 systemd[1]: Mounted /media/samba.
Oct 27 23:39:52 dev1 systemd[1]: Unmounting /media/samba...
Oct 27 23:39:52 dev1 systemd[1]: Unmounted /media/samba.
[root@dev1 ~]# systemctl restart media-samba.mount
[root@dev1 ~]# systemctl status media-samba.mount
media-samba.mount - /media/samba
   Loaded: loaded (/etc/fstab)
   Active: active (mounted) since Mon 2014-10-27 23:40:32 ART; 2s ago
    Where: /media/samba
     What: //192.168.0.10/gacanepa
  Process: 2411 ExecUnmount=/bin/umount /media/samba (code=exited, status=0/SUCCESS)
  Process: 2418 ExecMount=/bin/mount //192.168.0.10/gacanepa /media/samba -t cifs -o credentials=/media/samba/.smb

Oct 27 23:40:32 dev1 systemd[1]: Mounting /media/samba...
Oct 27 23:40:32 dev1 systemd[1]: Mounted /media/samba.
[root@dev1 ~]#
```

Enabling or disabling a service to start during boot

Under systemd you can enable or disable a service when it boots.

```
systemctl enable [service] # enable a service
systemctl disable [service] # prevent a service from starting at boot
```

The process of enabling or disabling a service to start automatically on boot consists in adding or removing symbolic links in the /etc/systemd/system/multi-user.target.wants directory:

```
[root@dev1 ~]# systemctl enable ebtables.service
ln -s '/usr/lib/systemd/system/ebtables.service' '/etc/systemd/system/multi-user.target.wants/ebtables.service'
[root@dev1 ~]# ls -l /etc/systemd/system/multi-user.target.wants
total 0
lrwxrwxrwx. 1 root root 38 Sep 18 20:55 auditd.service -> /usr/lib/systemd/system/auditd.service
lrwxrwxrwx. 1 root root 44 Sep 18 20:54 avahi-daemon.service -> /usr/lib/systemd/system/avahi-daemon.service
lrwxrwxrwx. 1 root root 37 Sep 18 20:54 crond.service -> /usr/lib/systemd/system/crond.service
lrwxrwxrwx. 1 root root 40 Oct 28 00:04 ebtables.service -> /usr/lib/systemd/system/ebtables.service
lrwxrwxrwx. 1 root root 42 Sep 18 20:55 irqbalance.service -> /usr/lib/systemd/system/irqbalance.service
```

```
[root@dev1 ~]# systemctl disable ebtables.service
rm '/etc/systemd/system/multi-user.target.wants/ebtables.service'
[root@dev1 ~]# ls -l /etc/systemd/system/multi-user.target.wants
total 0
lrwxrwxrwx. 1 root root 38 Sep 18 20:55 auditd.service -> /usr/lib/systemd/system/auditd.service
lrwxrwxrwx. 1 root root 44 Sep 18 20:54 avahi-daemon.service -> /usr/lib/systemd/system/avahi-daemon.service
lrwxrwxrwx. 1 root root 37 Sep 18 20:54 crond.service -> /usr/lib/systemd/system/crond.service
lrwxrwxrwx. 1 root root 42 Sep 18 20:55 irqbalance.service -> /usr/lib/systemd/system/irqbalance.service
lrwxrwxrwx. 1 root root 37 Sep 18 20:54 kdump.service -> /usr/lib/systemd/system/kdump.service
lrwxrwxrwx. 1 root root 41 Oct 24 01:09 mdmonitor.service -> /usr/lib/systemd/system/mdmonitor.service
```

Alternatively, you can find out a service's current status (enabled or disabled) with the command

```
systemctl is-enabled [service]
```

For example,

```
systemctl is-enabled postfix.service
```

In addition, you can reboot the system with

```
systemctl reboot
```

or shut it down with

```
systemctl shutdown
```

Upstart

Upstart is an event-based replacement for the **/sbin/init** daemon and was born out of the need for starting services only, when they are needed (also supervising them while they are running), and handling events as they occur, thus surpassing the classic, dependency-based sysvinit system. It was originally developed for the Ubuntu distribution, but is used in Red Hat Enterprise Linux 6.0. Though it was intended to be suitable for deployment in all Linux distributions as a replacement for **sysvinit**, in time it was overshadowed by **systemd**. On February 14, 2014, Mark Shuttleworth (founder of Canonical Ltd.) announced that future releases of Ubuntu would use systemd as the default init daemon. Because the **SysV** startup script for system has been so common for so long, a large number of software packages include SysV startup scripts. To accommodate such packages, Upstart provides a compatibility mode: It runs SysV startup scripts in the usual locations (**/etc/rc.d/rc?.d**, **/etc/init.d/rc?.d**, **/etc/rc?.d**, or a similar location). Thus, if we install a package that doesnâ€™t yet include an Upstart configuration script, it should still launch in the usual way. Furthermore, if we have installed utilities such as chkconfig, you should be able to use them to manage your SysV-based services just as we would on sysvinit based systems. Upstart scripts also support starting or stopping services based on a wider variety of

actions than do SysV startup scripts; for example, Upstart can launch a service whenever a particular hardware device is attached. A system that uses Upstart and its native scripts exclusively replaces the **/etc/inittab** file and the runlevel-specific **SysV** startup script directories with **.conf** scripts in the **/etc/init** directory. These ***.conf** scripts (also known as job definitions) generally consists of the following:

1. Description of the process.
2. Runlevels where the process should run or events that should trigger it.
3. Runlevels where process should be stopped or events that should stop it.
4. Options.
5. Command to launch the process.

For example,

```
# My test service - Upstart script demo description "Here goes the description of 'My
test service'" author "Dave Null <dave.null@example.com>"
# Stanzas

#
# Stanzas define when and how a process is started and stopped
# See a list of stanzas here: http://upstart.ubuntu.com/wiki/Stanzas#respawn
# When to start the service
start on runlevel [2345]
# When to stop the service
stop on runlevel [016]
# Automatically restart process in case of crash
respawn
# Specify working directory
chdir /home/dave/myfiles
# Specify the process/command (add arguments if needed) to run
exec bash backup.sh arg1 arg2
```

To apply changes, you will need to tell upstart to reload its configuration.

```
# initctl reload-configuration
```

Then start your job by typing the following command.

```
$ sudo start yourjobname
```

Where **yourjobname** is the name of the job that was added earlier with the **yourjobname.conf** script. A more complete and detailed reference guide for Upstart is available in the project's web site under the menu "Cookbook".

Summary

A knowledge of the Linux boot process is necessary to help you with troubleshooting tasks as well as with adapting the computer's performance and running services to your needs. In this article we have analyzed what happens from the moment when you press the Power switch to turn on the machine until you get a fully operational user interface.

Chapter 8: User management, special attributes, and PAM

Since Linux is a multi-user operating system (in that it allows multiple users on different computers or terminals to access a single system), you will need to know how to perform effective user management: how to add, edit, suspend, or delete user accounts, along with granting them the necessary permissions to do their assigned tasks.

Adding user accounts

To add a new user account, you can run either of the following two commands as root:

```
adduser [new_account]
useradd [new_account]
```

When a new user account is added to the system, the following operations are performed:

1) His/her home directory is created (/home/username by default).

2) The following hidden files are copied into the user's home directory, and will be used to provide environment variables for his/her user session.

```
.bash_logout
.bash_profile
.bashrc
```

3) A mail spool is created for the user.

4) A group is created and given the same name as the new user account.

The full account information is stored in the /etc/passwd file. This file contains a record per system user account and has the following format (fields are delimited by a colon):

```
[username]:[x]:[UID]:[GID]:[Comment]:[Home directory]:[Default shell]
```

1. Fields [username] and [Comment] are self explanatory.
2. The x in the second field indicates that the account is protected by a shadowed password (in /etc/shadow), which is needed to logon as [username].
3. The [UID] and [GID] fields are integers that represent the User IDentification and the primary Group IDentification to which [username] belongs, respectively.

Finally,

- the [Home directory] indicates the absolute path to [username]'s home directory, and
- [Default shell] is the shell that will be made available to this user when he or she logins the system.

Group information is stored in the /etc/group file. Each record has the following format:

```
[Group name]:[Group password]:[GID]:[Group members]
```

where

- [Group name] is the name of group.
- an x in [Group password] indicates group passwords are not being used.
- [GID]: same as in /etc/passwd
- [Group members]: a comma separated list of users who are members of [Group name].

```
[gacanepa@dev1 ~]$ grep gacanepa /etc/passwd
gacanepa:x:1000:1000:Gabriel Cánepa:/home/gacanepa:/bin/bash
[gacanepa@dev1 ~]$ grep gacanepa /etc/group
gacanepa:x:1000:gacanepa
[gacanepa@dev1 ~]$
```

After adding an account, you can edit the following information (to name a few fields) using the usermod command, whose basic syntax of usermod is as follows: **usermod [options] [username]**.

- To set the expiry date for an account, use the **--expiredate** flag followed by a date in YYYY-MM-DD format.
- To add the user to supplementary groups, use the combined **-aG**, or **--append --groups** options, followed by a comma separated list of groups.
- To change the default location of the user's home directory, use the **-d**, **or --home** options, followed by the absolute path to the new home directory.
- To change the shell the user will use by default. Use **--shell**, followed by the path to the new shell.
- To view the groups an user is a member of, do:

```
groups [username]
id [username]
```

```
[root@dev1 ~]# adduser tecmint
[root@dev1 ~]# usermod --expiredate 2014-10-30 --append --groups root,users --home /tmp --shell
[root@dev1 ~]# finger tecmint
Login: tecmint                          Name:
Directory: /tmp                         Shell: /bin/sh
Never logged in.
No mail.
No Plan.
[root@dev1 ~]# groups tecmint
tecmint : tecmint root users
[root@dev1 ~]# id tecmint
uid=1001(tecmint) gid=1001(tecmint) groups=1001(tecmint),0(root),100(users)
[root@dev1 ~]#
```

The finger command is used to look up information

The groups utility prints the names of the groups an user is in, when command also prints the corresponding UID and GIDs of those group

In the example above, we will set the expiry date of the tecmint user account to October 30th, 2014. We will also add the account to the root and users group. Finally, we will set sh as its default shell and change the location of the home directory to /tmp:

```
usermod --expiredate 2014-10-30 --append --groups root,users --home /tmp --shell
/bin/sh tecmint
```

For existing accounts, we can also do the following:

1. Disabling account by locking password: use the **--l** (lowercase **L**) or the **--lock** option to lock a user's password.
2. Unlocking password: use the **--u** or the **--unlock** option to unlock a user's password that was previously blocked.

```
usermod --lock tecmint
usermod --unlock tecmint
```

```
[root@dev1 ~]# usermod --lock tecmint ——> root locks the password for user tecmint
[root@dev1 ~]# exit ——> root logs off
logout
[gacanepa@dev1 ~]$ su tecmint ——> User gacanepa tries to logon as tecmint. Since the
Password:                         password is locked, the authentication fails
su: Authentication failure
[gacanepa@dev1 ~]$ su - ——> User gacanepa logins as root
Password:
Last login: Tue Oct 28 13:38:35 ART 2014 on pts/0
[root@dev1 ~]# usermod --unlock tecmint ——> root unlocks the password for user tecmint
[root@dev1 ~]# exit ——> root logs off
logout
[gacanepa@dev1 ~]$ su tecmint ——> User gacanepa logins as tecmint
Password:
sh-4.2$ ——> The authentication succeeds and a command prompt is shown
```

3. Creating a new group for read and write access to files that need to be accessed by several users

```
groupadd common_group # Add a new group
chmod :common_group common.txt # Change the group owner of common.txt to
common_group
usermod -aG common_group user1 # Add user1 to common_group
usermod -aG common_group user2 # Add user2 to common_group
usermod -aG common_group user3 # Add user3 to common_group
```

4. Deleting a group

You can delete a group with the following command:

```
groupdel [group_name]
```

If there are files owned by group_name, they will not be deleted, but the group owner will be set to the GID of the group that was deleted.

Deleting user accounts

You can delete an account (along with its home directory, if it's owned by the user, and all the files residing therein, and also the mail spool) using the userdel command with the --remove option:

userdel --remove [username]

Group management

Every time a new user account is added to the system, a group with the same name is created with the username as its only member. Other users can be added to the group later. One of the purposes of groups is to implement a simple access control to files and other system resources by setting the right permissions on those resources.

For example, suppose you have the following users

- user1 (primary group: user1)
- user2 (primary group: user2)
- user3 (primary group: user3)

All of them need read and write access to a file called common.txt located somewhere on your local system, or maybe on a network share that user1 has created. You may be tempted to do something like

```
chmod 660 common.txt
```

or

```
chmod u=rw,g=rw,o= common.txt # [notice the space between the last equal sign and the file name]
```

However, this will only provide read and write access to the owner of the file and to those users who are members of the group owner of the file (user1 in this case). Again, you may be tempted to add user2 and user3 to group user1, but that will also give them access to the rest of the files owned by user user1 and group user1.

This is where groups come in handy, and here's what you should do in a case like this.

Special permissions

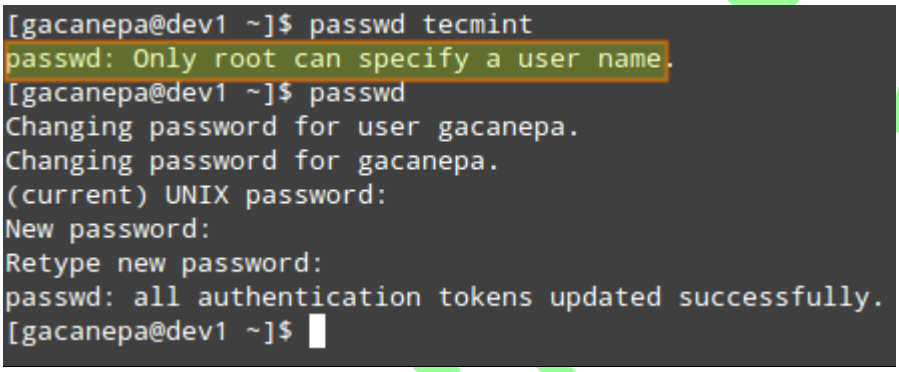
Besides the basic read, write, and execute permissions that we discussed in Chapter 3 ("Archiving and compression tools / File basic permissions and attributes") of this book, there are other less used (but not less important) permission settings, sometimes referred to as "special permissions". Like the basic permissions discussed earlier, they are set using an octal file or through a letter (symbolic notation) that indicates the type of permission.

SETUID

When the setuid permission is applied to an executable file, a user running the program inherits the effective privileges of the program's owner. Since this approach can reasonably raise security concerns, the number of files with setuid permission must be kept to a minimum. You will likely find programs with this permission set when a system user needs to access a file owned by root. Summing up, it isn't just that the user can execute the binary file, but also that he can do so with root's privileges.

For example, let's check the permissions of /bin/passwd. This binary is used to change the password of an account, and modifies the /etc/shadow file. The superuser can change anyone's password, but all other users should only be able to change their own.

Thus, any user should have permission to run /bin/passwd, but only root will be able to specify an account. Other users can only change their corresponding passwords.



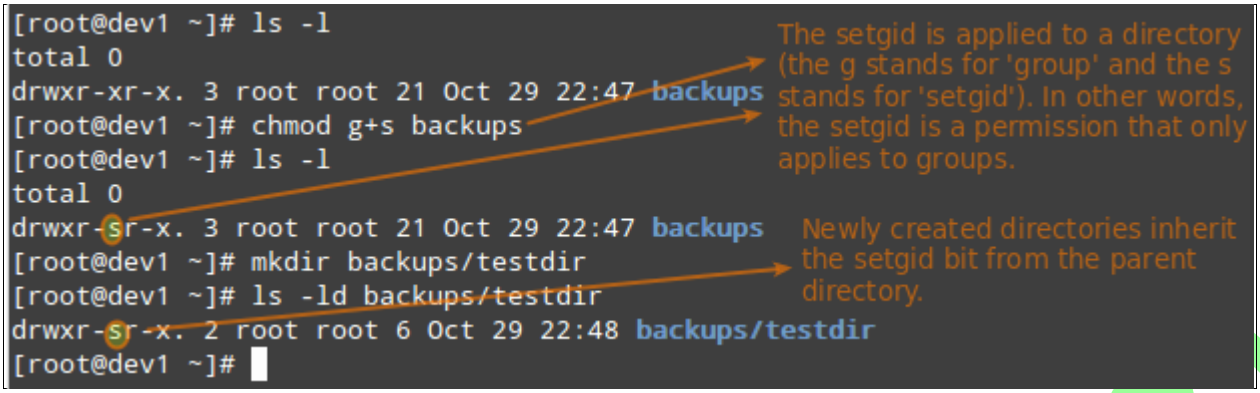
SETGID

When the setgid bit is set, the effective GID of the real user becomes that of the group owner. Thus, any user can access a file under the privileges granted to the group owner of such file. In addition, when the setgid bit is set on a directory, newly created files inherit the same group as the directory, and newly created subdirectories will also inherit the setgid bit of the parent directory. You will most likely use this approach whenever members of a certain group need access to all the files in a directory, regardless of the file owner's primary group.

```
chmod g+s [filename]
```

To set the setgid in octal form, prepend the number 2 to the current (or desired) basic permissions.

```
chmod 2755 [directory]
```



STICKY BIT

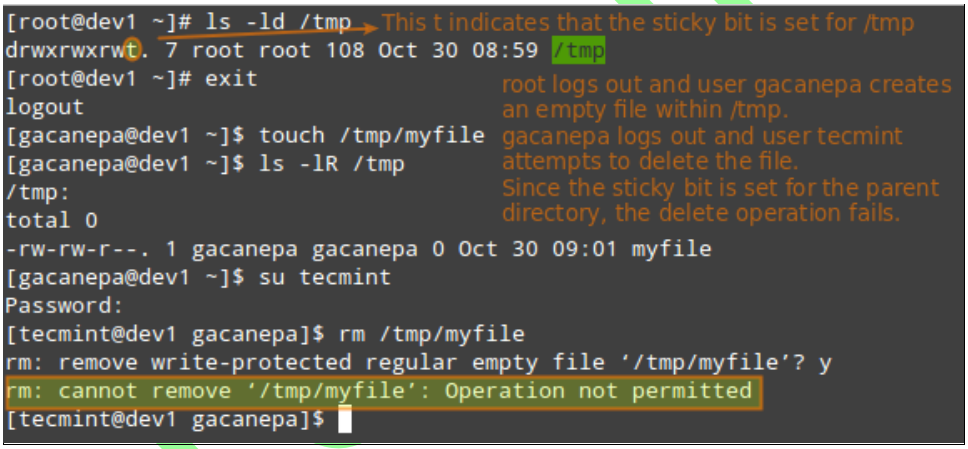
When the "sticky bit" is set on files, Linux just ignores it, whereas for directories it has the effect of preventing users from deleting or even renaming the files it contains unless the user owns the directory, the file, or is root.

```
chmod o+t [directory]
```

To set the sticky bit in octal form, prepend the number 1 to the current (or desired) basic permissions.

```
chmod 1755 [directory]
```

Without the sticky bit, anyone able to write to the directory can delete or rename files. For that reason, the sticky bit is commonly found on directories, such as /tmp, that are world-writable.

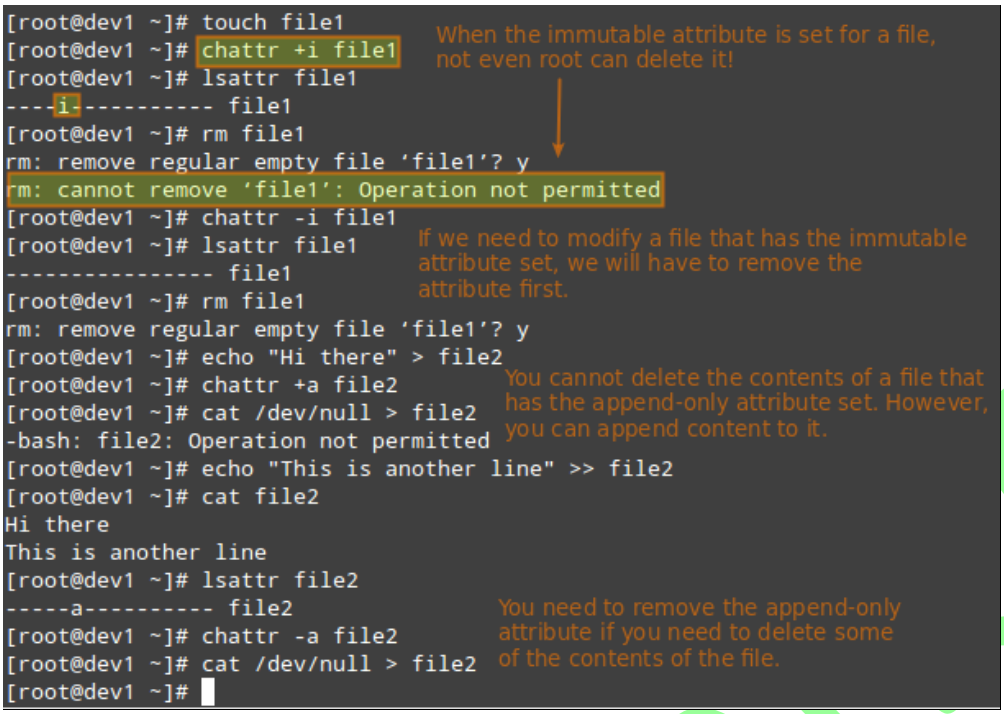


Special file attributes

There are other attributes that enable further limits on the operations that are allowed on files. For example, prevent the file from being renamed, moved, deleted, or even modified. They are set with the chattr command and can be viewed using the lsattr tool, as follows:

```
chattr +i file1
chattr +a file2
```

After executing those two commands, file1 will be immutable (which means it cannot be moved, renamed, modified or deleted) whereas file2 will enter append-only mode (can only be open in append mode for writing).



Accessing the root account and using sudo

One of the ways users can gain access to the root account is by typing

```
su
```

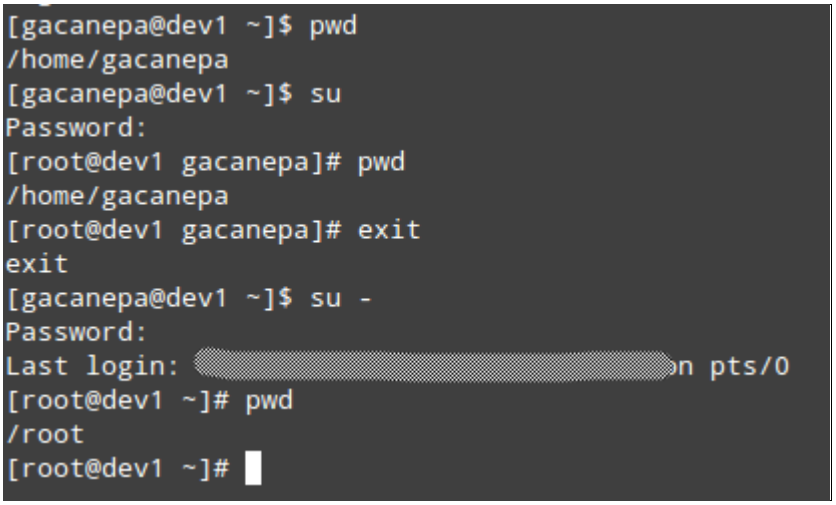
and then entering root's password.

If authentication succeeds, you will be logged on as root with the current working directory as the same as you were before.

If you want to be placed in root's home directory instead, run

```
su -
```

and then enter root's password.



The above procedure requires that a normal user knows root's password, which poses a serious security risk. For that reason, the sysadmin can configure the sudo command to allow an ordinary user to execute commands as a different user (usually the superuser) in a very controlled and limited way. Thus, restrictions can be set on a user so as to enable him to run one or more specific privileged commands and no others.

To authenticate using sudo, the user uses his/her own password. After entering the command, we will be prompted for our password (not the superuser's) and if the authentication succeeds (and if the user has been granted privileges to run the command), the specified command is carried out.

To grant access to sudo, the system administrator must edit the /etc/sudoers file. It is recommended that this file is edited using the visudo command instead of opening it directly with a text editor.

```
visudo
```

This opens the /etc/sudoers file using vim (you can follow the instructions given in Part 2 of this series to edit the file)

These are the most relevant lines:

```
Defaults    secure_path="/usr/sbin:/usr/bin:/sbin"
root        ALL=(ALL) ALL
tecmint     ALL=/bin/yum update
gacanepa    ALL=NOPASSWD:/bin/updatedb
%admin      ALL=(ALL) ALL
```

Let's take a closer look at them:

```
Defaults    secure_path="/usr/sbin:/usr/bin:/sbin:/usr/local/bin"
```

This line lets you specify the directories that will be used for sudo, and is used to prevent using user-specific directories, which can harm the system.

The next lines are used to specify permissions

```
root        ALL=(ALL) ALL
```

- The first ALL keyword indicates that this rule applies to all hosts.

- The second ALL indicates that the user in the first column can run commands with the privileges of any user.
- The third ALL means any command can be run.

```
tecmint      ALL=/bin/yum update
```

If no user is specified after the = sign, sudo assumes the root user. In this case, user tecmint will be able to run yum update as root.

```
gacanepa     ALL=NOPASSWD:/bin/updatedb
```

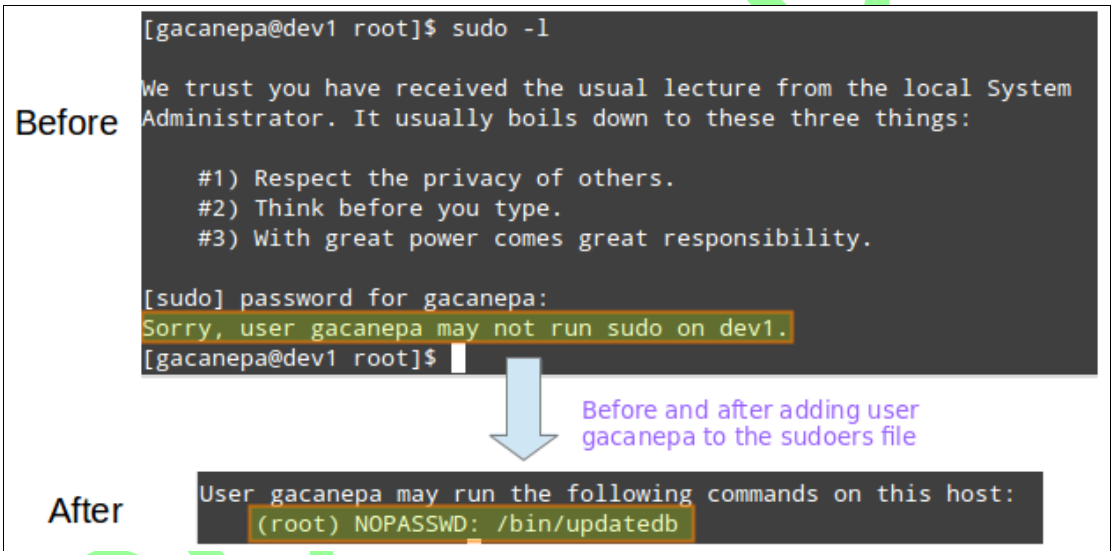
The NOPASSWD directive allows user gacanepa to run /bin/updatedb without needing to enter his password.

Finally,

```
%admin       ALL=(ALL) ALL
```

The % sign indicates that this line applies to a group called "admin". The meaning of the rest of the line is identical to that of an regular user. This means that members of the group "admin" can run all commands as any user on all hosts.

To see what privileges are granted to you by sudo, use the "-l" option to list them:



PAM (Pluggable Authentication Modules)

Pluggable Authentication Modules (PAM) offer the flexibility of setting a specific authentication scheme on a per-application and / or per-service basis using modules. This tool – present on all modern Linux distributions- overcame the problem often faced by developers in the early days of Linux, when each program that required authentication had to be compiled specially to know how to get the necessary information.

For example, with PAM, it doesn't matter whether your password is stored in /etc/shadow or on a separate server inside your network. For example, when the login program needs to authenticate a user, PAM provides dynamically the library that contains the functions for the right authentication scheme. Thus, changing the authentication scheme for the login application (or any other program using PAM) is easy since it only involves editing a configuration file (most likely, a file named after the application, located inside /etc/pam.d, and less likely in /etc/pam.conf).

Files inside /etc/pam.d indicate which applications are using PAM natively. In addition, we can tell whether a certain application uses PAM by checking if it the PAM library (libpam) has been linked to it:

```
ldd $(which login) | grep libpam # login uses PAM
ldd $(which top) | grep libpam # top does not use PAM
```

```
[root@server ~]# ldd $(which login) | grep libpam
        libpam.so.0 => /lib64/libpam.so.0 (0x00007f158e7a5000)
        libpam_misc.so.0 => /lib64/libpam_misc.so.0 (0x00007f158e5a1000)
[root@server ~]# ldd $(which top) | grep libpam
[root@server ~]#
```

In the above image we can see that the libpam has been linked with the **login** application. This makes sense since this application is involved in the operation of system user authentication, whereas **top** does not.

Let's examine the PAM configuration file for **passwd** – yes, the well-known utility to change user's passwords. It is located at /etc/pam.d/passwd:

```
[root@server ~]# cat /etc/pam.d/passwd
#%PAM-1.0
auth       include      system-auth
account    include      system-auth
password   substack     system-auth
-password   optional    pam_gnome_keyring.so use_authtok
password   substack     postlogin
[root@server ~]#
```

- The first column indicates the **type** of authentication to be used with the **module-path** (third column). When a hyphen appears before the type, PAM will not record to the system log if the module cannot be loaded because it could not be found in the system.

  The following authentication types are available:

  - **account**: this module type checks if the user or service has supplied valid credentials to authenticate.
  - **auth**: this module type verifies that the user is who he / she claims to be and grants any needed privileges.
  - **password**: this module type allows the user or service to update their password.
  - **session**: this module type indicates what should be done before and/or after the authentication succeeds.
- The second column (called **control**) indicates what should happen if the authentication with this module fails:
  - **requisite**: if the authentication via this module fails, overall authentication will be denied immediately.
  - **required** is similar to requisite, although all other listed modules for this service will be called before denying authentication.

  - **sufficient**: if the authentication via this module fails, PAM will still grant authentication even if a previous marked as required failed.
  - **optional**: if the authentication via this module fails or succeeds, nothing happens unless this is the only module of its type defined for this service.
  - **include** means that the lines of the given type should be read from another file.
  - **substack** is similar to includes but authentication failures or successes do not cause the exit of the complete module, but only of the substack.
- The fourth column, if it exists, shows the arguments to be passed to the module.

The first three lines in /etc/pam.d/passwd (shown above), load the system-auth module to check that the user has supplied valid credentials (**account**). If so, it allows him / her to change the authentication token (**password**) by giving permission to use passwd (**auth**).

For example, if you append

```
remember=2
```

to the following line

```
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass
use_authtok
```

in /etc/pam.d/system-auth:

```
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass
use_authtok remember=2
```

the last two hashed passwords of each user are saved in /etc/security/opasswd so that they cannot be reused:

For more information refer to the Linux-PAM System Administrator's guide and in `man 5 pam.conf`.

Summary

Effective user and file management skills are essential tools for any system administrator. In this article we have covered the basics and hope you can use it as a good starting to point to build upon.

Chapter 9: Package management

In few words, package management is a method of installing and maintaining (which includes updating and probably removing as well) software on the system.

In the early days of Linux, programs were only distributed as source code, along with the required man pages, the necessary configuration files, and more. Nowadays, most Linux distributors use by default prebuilt programs or sets of programs called packages, which are presented to users ready for installation on that distribution. However, one of the wonders of Linux is still the possibility to obtain source code of a program to be studied, improved, and compiled.

How package management systems work

If a certain package requires a certain resource -such as a shared library, or another package-, it is said to have a dependency. All modern package management systems provide some method of dependency resolution to ensure that when a package is installed, all of its dependencies are installed as well.

Packaging Systems

Almost all the software that is installed on a modern a Linux system will be found on the Internet. It can either be provided by the distribution vendor through central repositories (which can contain several thousands of packages, each of which has been specifically built, tested, and maintained for the distribution) or be available in source code that can be installed downloaded and installed manually.

Because different distribution families use different packaging systems (Debian: *.deb / CentOS: *.rpm / openSUSE: *.rpm built specially for openSUSE), a package intended for one distribution will not be compatible with another distribution. However, most distributions are likely to fall into one of the three distribution families covered by the LFCS certification.

High And Low-level Package Tools

In order to perform the task of package management effectively, you need to be aware that you will have available two types of utilities: low-level tools (which handle in the backend the actual installation, upgrade, and removal of package files), and high-level tools (which are in charge of ensuring that the tasks of dependency resolution and metadata searching -”data about the data”- are performed).

DISTRIBUTION	LOW-LEVEL TOOL	HIGH-LEVEL TOOL
Debian and derivatives	dpkg	apt-get / aptitude
CentOS	rpm	yum
openSUSE	rpm	zypper

The most frequent tasks that you will do with low level tools are as follows:

1) Installing a package from a compiled (*.deb or *.rpm) file

The downside of this installation method is that no dependency resolution is provided. You will most likely choose to install a package from a compiled file when such package is not available in the distribution's repositories and therefore cannot be downloaded and installed through a high-level tool. Since low-level tools do not perform dependency resolution, they will exit with an error if we try to install a package with unmet dependencies.

```
dpkg -i file.deb # Debian and derivatives
rpm -Uvh file.rpm # CentOS / openSUSE
```

One small caveat. Do not attempt to install on CentOS a *.rpm file that was built for openSUSE, or viceversa!

2) Upgrading a package from a compiled file

Again, you will only upgrade an installed package manually when it is not available in the central repositories.

```
dpkg -i file.deb # Debian and derivatives
rpm -Uvh file.rpm # CentOS / openSUSE
```

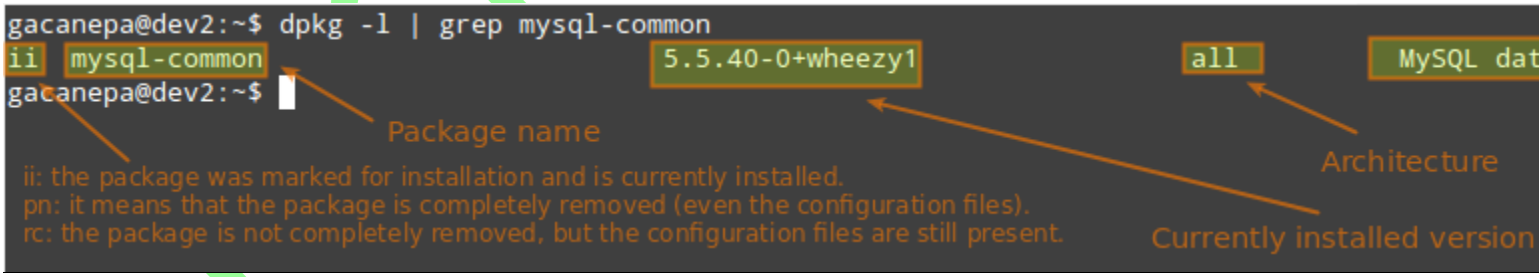
3) Listing installed packages

When you first get your hands on an already working system, chances are you'll want to know what packages are installed:

```
dpkg -l # Debian and derivatives
rpm -qa # CentOS / openSUSE
```

If you want to know whether a specific package is installed, you can pipe the output of the above commands to grep, as explained in Part 1 of this series. Suppose we need to verify if package mysql-common is installed on an Ubuntu system:

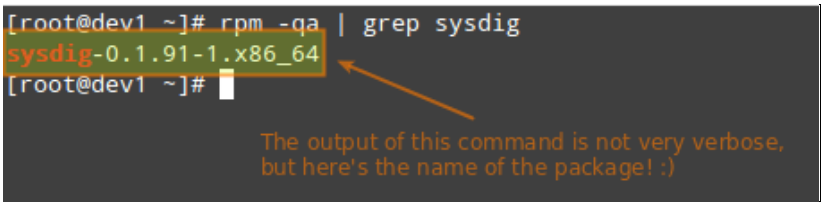
```
dpkg -l | grep mysql-common
```



Another way to determine if a package is installed:

```
dpkg --status package_name # Debian and derivatives
rpm -q package_name # CentOS / openSUSE
```

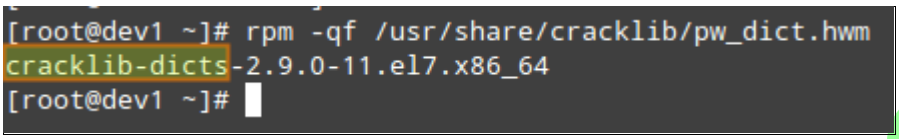
For example, let's find out whether package **sysdig** is installed on our system



4) Finding out which package installed a file

```
dpkg --search file_name# Debian and derivatives
rpm -qf file_name # CentOS / openSUSE
```

For example, which package installed pw_dict.hwm?



Common usage of high-level tools

1) Searching for a package

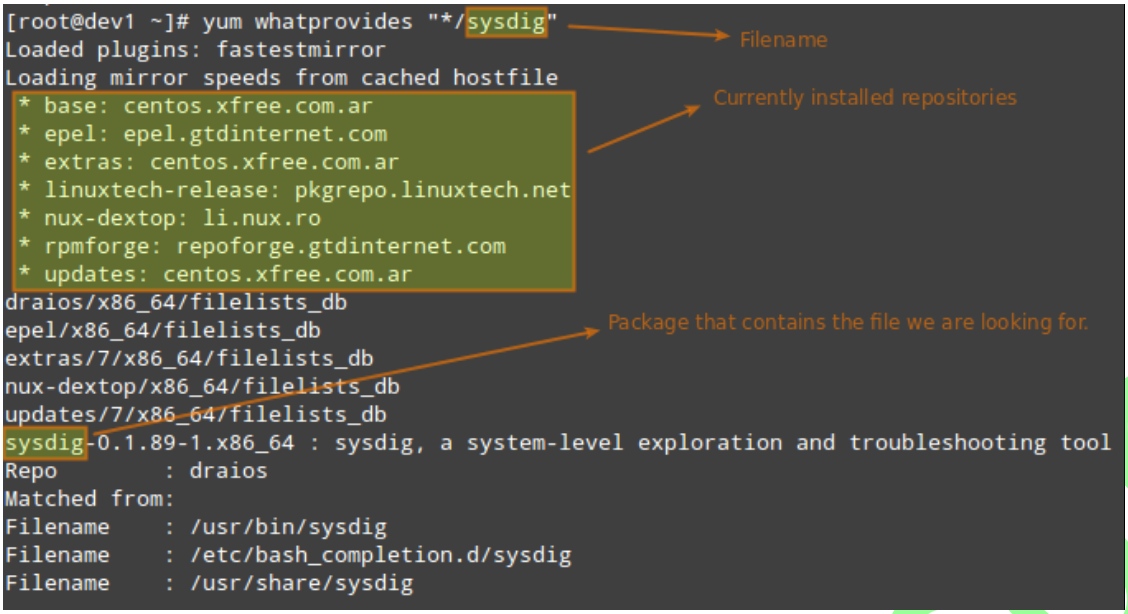
```
aptitude update && aptitude search package_name # Debian and derivatives
yum search package_name # CentOS / openSUSE
yum search all package_name # CentOS / openSUSE
```

In the search all option, yum will search for package_name not only in package names, but also in package descriptions.

```
yum whatprovides "*/package_name"  # CentOS
```

Let's supposed we need a file whose name is sysdig. To know that package we will have to install, let's run

```
yum whatprovides "*/sysdig"
```



whatprovides tells yum to search the package the will provide a file that matches the above regular expression.

```
zypper refresh && zypper search package_name # openSUSE
```

Installing a package from a repository

While installing a package, you may be prompted to confirm the installation after the package manager has resolved all dependencies. Note that running update or refresh (according to the package manager being used) is not strictly necessary, but keeping installed packages up to date is a good sysadmin practice for security and dependency reasons.

```
aptitude update && aptitude install package_name # Debian and derivatives
yum update && yum install package_name # CentOS
zypper refresh && zypper install package_name # openSUSE
```

Removing a package

Most (if not all) package managers will prompt you, by default, if you're sure about proceeding with the uninstallation before actually performing it. So read the onscreen messages carefully to avoid running into unnecessary trouble!

```
aptitude remove / purge package_name # Debian and derivatives
```

remove will uninstall the package but leaving configuration files intact, whereas **purge** will erase every trace of the program from your system.

```
yum erase package_name # CentOS
zypper install -package_name # Notice the minus sign in front of the package that will be uninstalled, openSUSE
```

Displaying information about a package

```
aptitude show package_name # Debian and derivatives
```

The following command will display information about the birthday package:

```
aptitude show birthday # Debian and derivatives
```

```
gacanepa@dev2:~$ aptitude show birthday
Package: birthday
New: yes
State: not installed
Version: 1.6.2-3
Priority: optional
Section: misc
Maintainer: Bart Martens <bartm@debian.org>
Architecture: i386
Uncompressed Size: 69.6 k
Depends: libc6 (>= 2.7-1)
Recommends: perl
Description: Display information about pending events on login
 Given a list of the dates of various different events, works out and displays a list of th
 designed for birthdays, but can equally be used for reminders about yearly events, or for

 Note that if you want to use vcf2birthday script you will need perl.
Homepage: http://sourceforge.net/projects/birthday/

gacanepa@dev2:~$
```

```
yum info package_name # CentOS
zypper info package_name # openSUSE
```

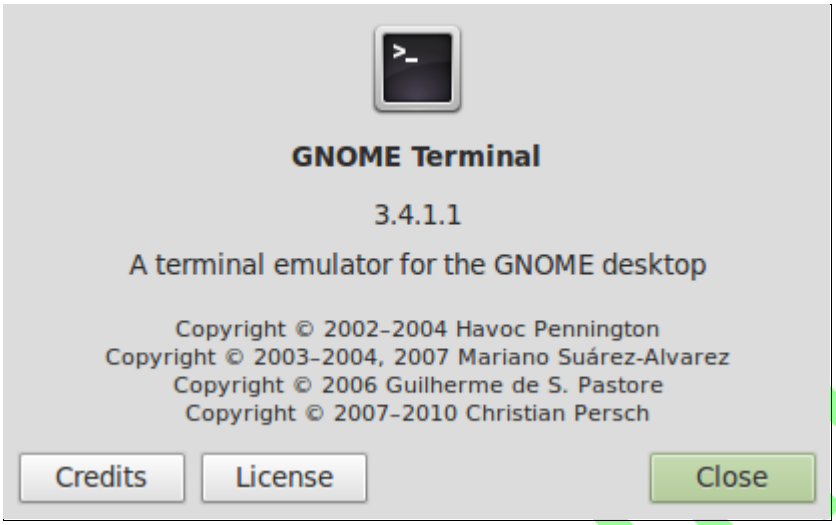
Summary

Package management is something you just can't sweep under the rug as a system administrator. You should be prepared to use the tools described in this article at a moment's notice. We hope you find it useful in your preparation for the LFCS exam and for your daily tasks.

Chapter 10: Terminals, shells, and filesystem troubleshooting

Let’s clarify a few concepts first:

- A shell is a program that accepts commands and gives them to the operating system to be executed.
- A terminal is a program that allows us as end users to interact with the shell. One example of a terminal is GNOME terminal, as shown in the below image:



When we first start a shell, it presents a command prompt (also known as the command line), which tells us that the shell is ready to start accepting commands from its standard input device.

Linux provides a range of options for shells, the following being the most common:

1) bash: Bash stands for Bourne Again SHell and is the GNU Project's default shell. In addition, it’s the one emphasized on the LPIC-1 exam. It incorporates useful features from the Korn shell (ksh) and C shell (csh), offering several improvements at the same time. This is the default shell used by the distributions covered in the LFCS certification, and it is the shell that we will use in this tutorial.

2) sh: The Bourne SHell is the oldest shell and therefore has been the default shell of many UNIX-like operating systems for many years.

3) ksh: The Korn SHell is a Unix shell which was developed by David Korn at Bell Labs in the early 1980s. It is backward-compatible with the Bourne shell and includes many features of the C shell.

A shell script is nothing more and nothing less than a text file turned into an executable program that combines commands that are executed by the shell one after another.

Basic shell scripting

As mentioned earlier, a shell script is born as a plain text file. Thus, can be created and edited using our preferred text editor. You may want to consider using vi/m, which features syntax highlighting for your convenience.

Type

```
vim myscript.sh
```

and press Enter.

The very first line of a shell script must be as follows (also known as a shebang):

```
#!/bin/bash
```

It "tells" the operating system the name of the interpreter that should be used to run the text that follows.

Now it's time to add our commands. We can clarify the purpose of each command, or the entire script, by adding comments as well. Note that the shell ignores those lines beginning with a pound sign # (explanatory comments).

```
#!/bin/bash
echo This is Part 10 of the 10-article series about the LFCS certification
echo Today is $(date +%Y-%m-%d)
```

Once the script has been written and saved, we need to make it executable:

```
chmod 755 myscript.sh
```

Before running our script, we need to say a few words about the $PATH environment variable. If we run

```
echo $PATH
```

from the command line, we will see the contents of $PATH: a colon-separated list of directories that are searched when we enter the name of a executable program. It is called an environment variable because it is part of the shell environment - a set of information that becomes available for the shell and its child processes when the shell is first started.

When we type a command and press Enter, the shell searches in all the directories listed in the $PATH variables and executes the first instance that is found. Let's see an example:

```
[gacanepa@dev1 ~]$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/gacanepa/.local/bin:/home/gacanepa/bin
[gacanepa@dev1 ~]$
```

If there are two executable files with the same name, one in /usr/local/bin and another in /usr/bin, the one in the first directory will be executed first, whereas the other will be disregarded.

If we haven't saved our script inside one of the directories listed in the $PATH variable, we need to prepend ./ to the file name in order to execute it. Otherwise, we can run it just as we would do with a regular command:

```
[gacanepa@dev1 scripts]$ pwd
/home/gacanepa/scripts
[gacanepa@dev1 scripts]$ ls
myscript.sh
[gacanepa@dev1 scripts]$ ./myscript.sh
This is Part 10 of the 10-article series about the LFCS certification
Today is 2014-11-05
[gacanepa@dev1 scripts]$ cp myscript.sh ../bin
[gacanepa@dev1 scripts]$ cd ../bin
[gacanepa@dev1 bin]$ pwd
/home/gacanepa/bin
[gacanepa@dev1 bin]$ myscript.sh
This is Part 10 of the 10-article series about the LFCS certification
Today is 2014-11-05
[gacanepa@dev1 bin]$
```

Our script is not located inside any of the directories listed in $PATH. For that reason, we need to precede the script name with ./ in order to run it.

If we copy myscript.sh into one of the directories that are listed in $PATH, we will be able to run the script without the leading ./ as in the last case.

Conditionals

Whenever you need to specify different courses of action to be taken in a shell script, as result of the success or failure of a command, you will use the **if** construct to define such conditions. Make sure there is a space between the semicolon and the **then** keyword! Its basic syntax is:

```
if CONDITION; then
COMMANDS;
else
OTHER-COMMANDS
fi
```

where CONDITION can be one of the following (only the most frequent conditions are cited here) and evaluates to true when:

- [-a file] → file exists.
- [-d file] → file exists and is a directory.
- [-f file] →file exists and is a regular file.
- [-u file] →file exists and its SUID (set user ID) bit is set.
- [-g file] →file exists and its SGID bit is set.
- [-k file] →file exists and its sticky bit is set.
- [-r file] →file exists and is readable.
- [-s file]→ file exists and is not empty.
- [-w file]→file exists and is writable.
- [-x file] is true if file exists and is executable.
- [string1 = string2] → the strings are equal.
- [string1 != string2] →the strings are not equal.
- [int1 op int2] where op is one of the following comparison operators:

- -eq --> is true if int1 is equal to int2.
- -ne --> true if int1 is not equal to int2.
- -lt --> true if int1 is less than int2.
- -le --> true if int1 is less than or equal to int2.
- -gt --> true if int1 is greater than int2.
- -ge --> true if int1 is greater than or equal to int2.

FOR LOOPS

This loop allows to execute one or more commands for each value in a list of values. Its basic syntax is:

```
for item in SEQUENCE; do
COMMANDS;
done
```

where item is a generic variable that represents each value in SEQUENCE during each iteration.

WHILE LOOPS

This loop allows to execute a series of repetitive commands as long as the control command executes with an exit status equal to zero (successfully). Its basic syntax is:

```
while EVALUATION_COMMAND; do
EXECUTE_COMMANDS;
done
```

where EVALUATION_COMMAND can be any command(s) that can exit with a success (0) or failure (other than 0) status, and EXECUTE_COMMANDS can be any program, script or shell construct, including other nested loops.

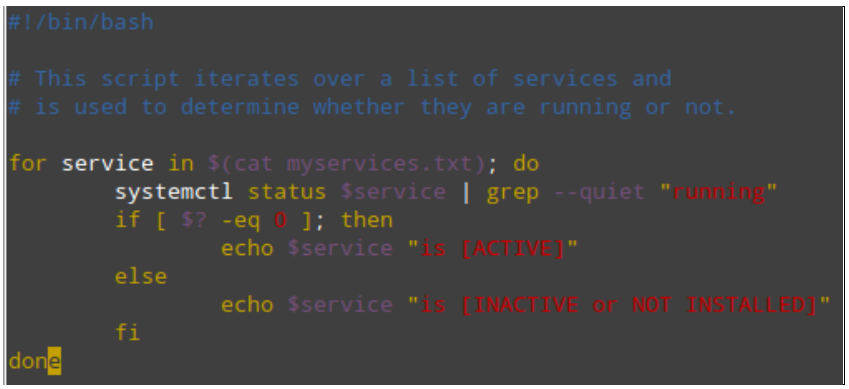
Putting it all together

We will demonstrate the use of the if construct and the for loop with the following example, which we will use to find out if a service is running in a systemd-based distro.

Let's create a file with a list of services that we want to monitor at a glance:

```
[gacanepa@dev1 ~]$ cat myservices.txt
sshd
mariadb
httpd
crond
firewalld
[gacanepa@dev1 ~]$
```

Our shell script should look like:



```
#!/bin/bash
# This script iterates over a list of services and
# is used to determine whether they are running or not.
for service in $(cat myservices.txt); do
        systemctl status $service | grep --quiet "running"
        if [ $? -eq 0 ]; then
                echo $service "is [ACTIVE]"
        else
                echo $service "is [INACTIVE or NOT INSTALLED]"
        fi
done
```

Let's explain how the script works:

1) The for loop reads the myservices.txt file one element of LIST at a time. That single element is denoted by the generic variable named service. The LIST is populated with the output of

```
cat myservices.txt
```

2) The above command is enclosed in parentheses and preceded by a dollar sign to indicate that it should be evaluated to populate the LIST that for will iterate over.

3) For each element of LIST (meaning every instance of the service variable), the following command will be executed:

```
systemctl status $service | grep --quiet "running"
```

Alternatively, we can replace the above line with

```
systemctl is-active $service | grep --quiet "active"
```

This time we need to precede our generic variable (which represents each element in LIST) with a dollar sign to indicate it's a variable and thus its value in each iteration should be used. The output is then piped to grep. The --quiet flag is used to prevent grep from displaying to the screen the lines where the word running appears. When that happens, the above command returns an exit status of 0 (represented by $? in the if construct), thus verifying that the service is running. An exit status different than 0 (meaning the word running was not found in the output of systemctl status $service) indicates that the service is not running.

```
[gacanepa@dev1 ~]$ ./for_demo.sh
sshd is [ACTIVE]
mariadb is [INACTIVE or NOT INSTALLED]
httpd is [INACTIVE or NOT INSTALLED]
crond is [ACTIVE]
firewalld is [ACTIVE]
[gacanepa@dev1 ~]$
```

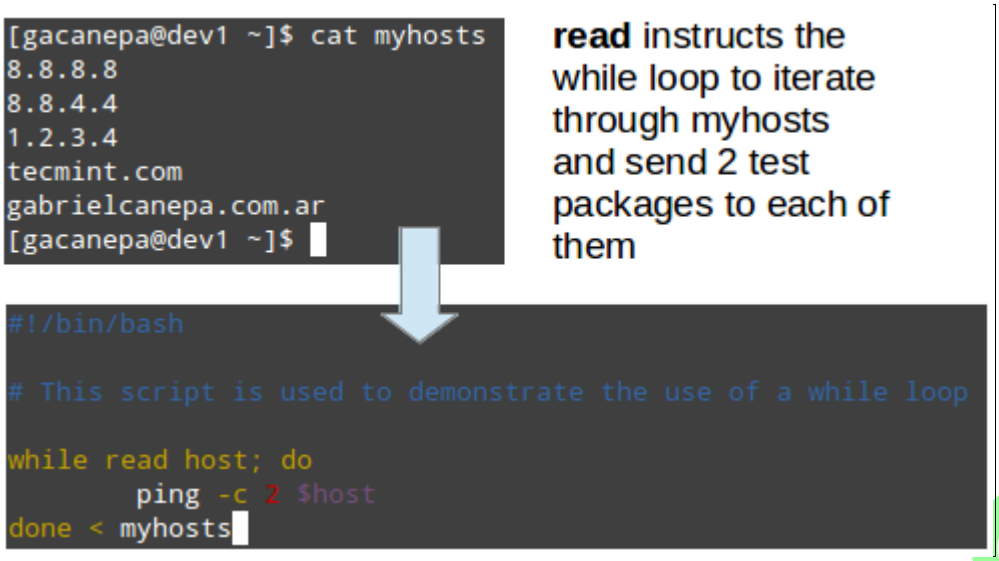
We could go one step further and check for the existence of myservices.txt before even attempting to enter the for loop

```
#!/bin/bash
# This script iterates over a list of services and
# is used to determine whether they are running or not.
if [ -f myservices.txt ]; then
            for service in $(cat myservices.txt); do
                    systemctl status $service | grep --quiet "running"
                    if [ $? -eq 0 ]; then
                            echo $service "is [ACTIVE]"
                    else
                            echo $service "is [INACTIVE or NOT INSTALLED]"
                    fi
            done
else
            echo "myservices.txt is missing"
fi
```

Pinging a series of network or internet hosts for reply statistics

You may want to maintain a list of hosts in a text file and use a script to determine every now and then whether they're pingable or not (feel free to replace the contents of myhosts and try for yourself). The read shell built-in command tells the while loop to read myhosts line by line and assigns the content of each line to variable host, which is then passed to the ping command.

```
#!/bin/bash
# This script is used to demonstrate the use of a while loop
while read host; do
            ping -c 2 $host
done < myhosts
```



Filesystem troubleshooting

Although Linux is a very stable operating system, if it crashes for some reason (for example, due to a power outage), one (or more) of your file systems will not be unmounted properly and thus will be automatically checked for errors when Linux is restarted.

In addition, each time the system boots during a normal boot, it always checks the integrity of the filesystems before mounting them. In both cases this is performed using a tool named fsck ("file system check"). fsck will not only check the integrity of file systems, but also attempt to repair corrupt file systems if instructed to do so. Depending on the severity of damage, fsck may succeed or not; when it does, recovered portions of files are placed in the lost+found directory, located in the root of each file system.

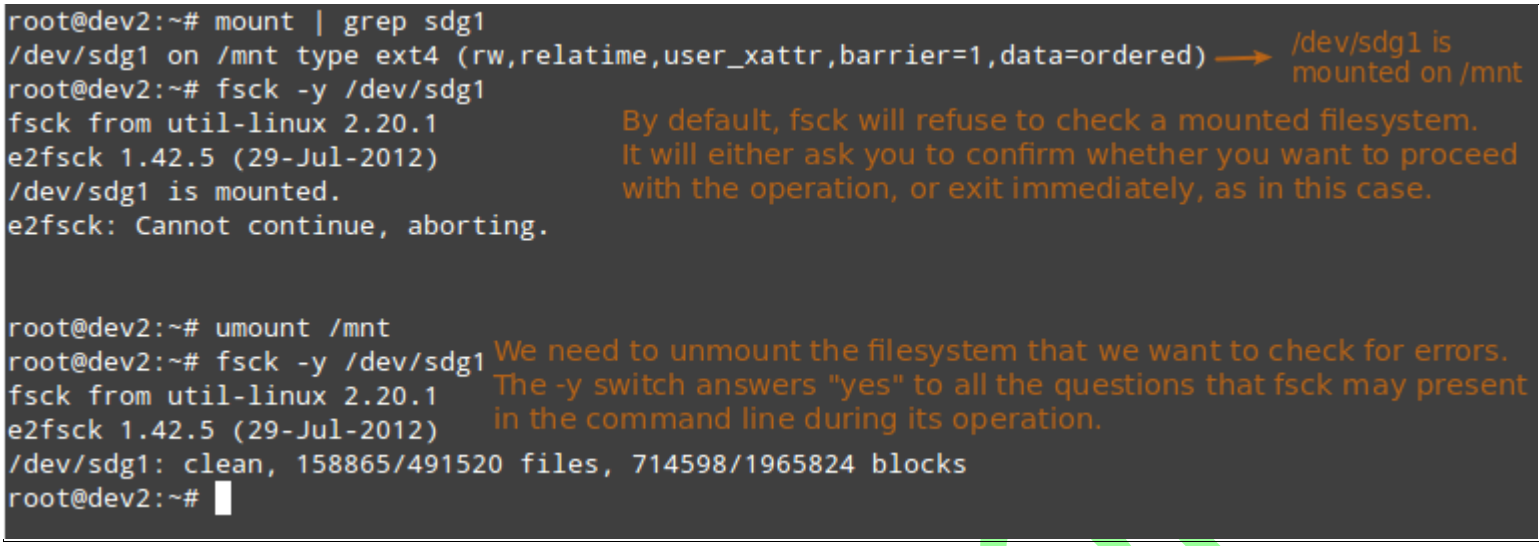
Last but not least, we must note that inconsistencies may also happen if we try to remove an USB drive when the operating system is still writing to it, and may even result in hardware damage.

The basic syntax of fsck is as follows:

```
fsck [options] filesystem
```

Checking a filesystem for errors and attempting to repair automatically

In order to check a filesystem with fsck, we must first unmount it:



Besides the -y flag, we can use the -a option to automatically repair the file systems without asking any questions, and force the check even when the filesystem looks clean:

```
fsck -af /dev/sdg1
```

If we're only interested in finding out what's wrong (without trying to fix anything for the time being) we can run fsck with the -n option, which will output the filesystem issues to standard output.

```
fsck -n /dev/sdg1
```

Depending on the error messages in the output of fsck, we will know whether we can try to solve the issue ourselves or escalate it to engineering teams to perform further checks on the hardware.

Summary

In this chapter we have explained the fundamentals of shell scripting and filesystem troubleshooting, which are valuable and essentials skills that every system administrator must have.

Chapter 11: Logical Volume Management

One of the most important decisions while installing a Linux system is the amount of storage space to be allocated for system files, home directories, and others. If you make a mistake at that point, growing a partition that has run out of space can be burdensome and somewhat risky.

Logical Volumes Management (also known as LVM), which have become a default for the installation of most (if not all) Linux distributions, have numerous advantages over traditional partitioning management. Perhaps the most distinguishing feature of LVM is that it allows logical divisions to be resized (reduced or increased) at will without much hassle.

The structure of the LVM consists of:

- One or more entire hard disks or partitions are configured as physical volumes (PVs).
- A volume group (VG) is created using one or more physical volumes. You can think of a volume group as a single storage unit.
- Multiple logical volumes can then be created in a volume group. Each logical volume is somewhat equivalent to a traditional partition - with the advantage that it can be resized at will as we mentioned earlier.

In this article we will use three disks of 8 GB each (/dev/sdb, /dev/sdc, and /dev/sdd) to create three physical volumes. You can either create the PVs directly on top of the device, or partition it first. Although we have chosen to go with the first method, if you decide to go with the second (as explained in Part 4 of this series) make sure to configure each partition as type 8e.

Creating physical volumes, volume groups, and logical volumes

To create physical volumes on top of /dev/sdb, /dev/sdc, and /dev/sdd, do:

```
pvcreate /dev/sdb /dev/sdc /dev/sdd
```

You can list the newly created PVs with

```
pvs
```

and get detailed information about each PV with

```
pvdisplay /dev/sdX
```

(where X is b, c, or d)

If you omit /dev/sdX as parameter, you will get information about all the PVs.

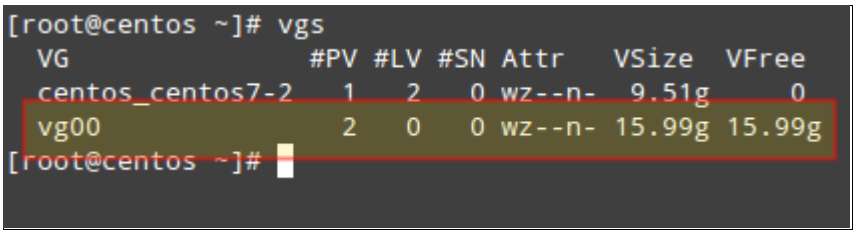
To create a volume group named vg00 using /dev/sdb and /dev/sdc (we will save /dev/sdd for later to illustrate the possibility of adding other devices to expand storage capacity when needed):

```
vgcreate vg00 /dev/sdb /dev/sdc
```

As it was the case with physical volumes, you can also view information about this volume group by issuing

```
vgdisplay vg00
```

Since vg00 is formed with two 8 GB disks, it will appear as a single 16 GB drive:



When it comes to creating logical volumes, the distribution of space must take into consideration both current and future needs. It is considered good practice to name each logical volume according to its intended use. For example, let's create two LVs named vol_projects (10 GB) and vol_backups (remaining space), which we can use later to store project documentation and system backups, respectively.

The -n option is used to indicate a name for the LV, whereas -L sets a fixed size and -l (lowercase L) is used to indicate a percentage of the remaining space in the container VG.

```
lvcreate -n vol_projects -L 10G vg00
lvcreate -n vol_backups -l 100%FREE vg00
```

As before, you can view the list of LVs and basic information with

```
lvs
```

and detailed information with

```
lvdisplay
```

To view information about a single LV, use lvdisplay with the VG and LV as parameters, as follows:

```
lvdisplay vg00/vol_projects
```

```
[root@centos ~]# lvdisplay vg00/vol_projects
  --- Logical volume ---
  LV Path                /dev/vg00/vol_projects
  LV Name                vol_projects
  VG Name                vg00
  LV UUID                03os1b-encP-XVDX-wAHY-j1LK-iAdf-l8WD1y
  LV Write Access        read/write
  LV Creation host, time centos, 2016-02-20 19:27:39 -0500
  LV Status              available
  # open                 0
  LV Size                10.00 GiB
  Current LE             2560
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:2

[root@centos ~]#
```

In the image above we can see that the LVs were created as storage devices (refer to the LV Path line). Before each logical volume can be used, we need to create a filesystem on top of it. We'll use ext4 as an example here since it allows us both to increase and reduce the size of each LV (as opposed to xfs that only allows to increase the size):

```
mkfs.ext4 /dev/vg00/vol_projects
mkfs.ext4 /dev/vg00/vol_backups
```

In the next section we will explain how to resize logical volumes and add extra physical storage space when the need arises to do so.

Resizing logical volumes and extending volume groups

Now picture the following scenario. You are starting to run out of space in vol_backups, while you have plenty of space available in vol_projects. Due to the nature of LVM, we can easily reduce the size of the latter (say 2.5 GB) and allocate it for the former, while resizing each filesystem at the same time.

Fortunately, this is as easy as doing:

```
lvreduce -L -2.5G -r /dev/vg00/vol_projects
lvextend -l +100%FREE -r /dev/vg00/vol_backups
```

```
[root@centos ~]# lvreduce -L -2.5G -r /dev/vg00/vol_projects
fsck from util-linux 2.23.2
/dev/mapper/vg00-vol_projects: clean, 11/655360 files, 83128/2621440 blocks
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /dev/mapper/vg00-vol_projects to 1966080 (4k) blocks.
The filesystem on /dev/mapper/vg00-vol_projects is now 1966080 blocks long.

  Size of logical volume vg00/vol_projects changed from 10.00 GiB (2560 extents) to 7.50 GiB (19
  Logical volume vol_projects successfully resized.
[root@centos ~]# lvextend -l +100%FREE -r /dev/vg00/vol_backups
fsck from util-linux 2.23.2
/dev/mapper/vg00-vol_backups: clean, 11/393216 files, 63590/1570816 blocks
  Size of logical volume vg00/vol_backups changed from 5.99 GiB (1534 extents) to 8.49 GiB (2174
  Logical volume vol_backups successfully resized.
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /dev/mapper/vg00-vol_backups to 2226176 (4k) blocks.
The filesystem on /dev/mapper/vg00-vol_backups is now 2226176 blocks long.

[root@centos ~]#
```

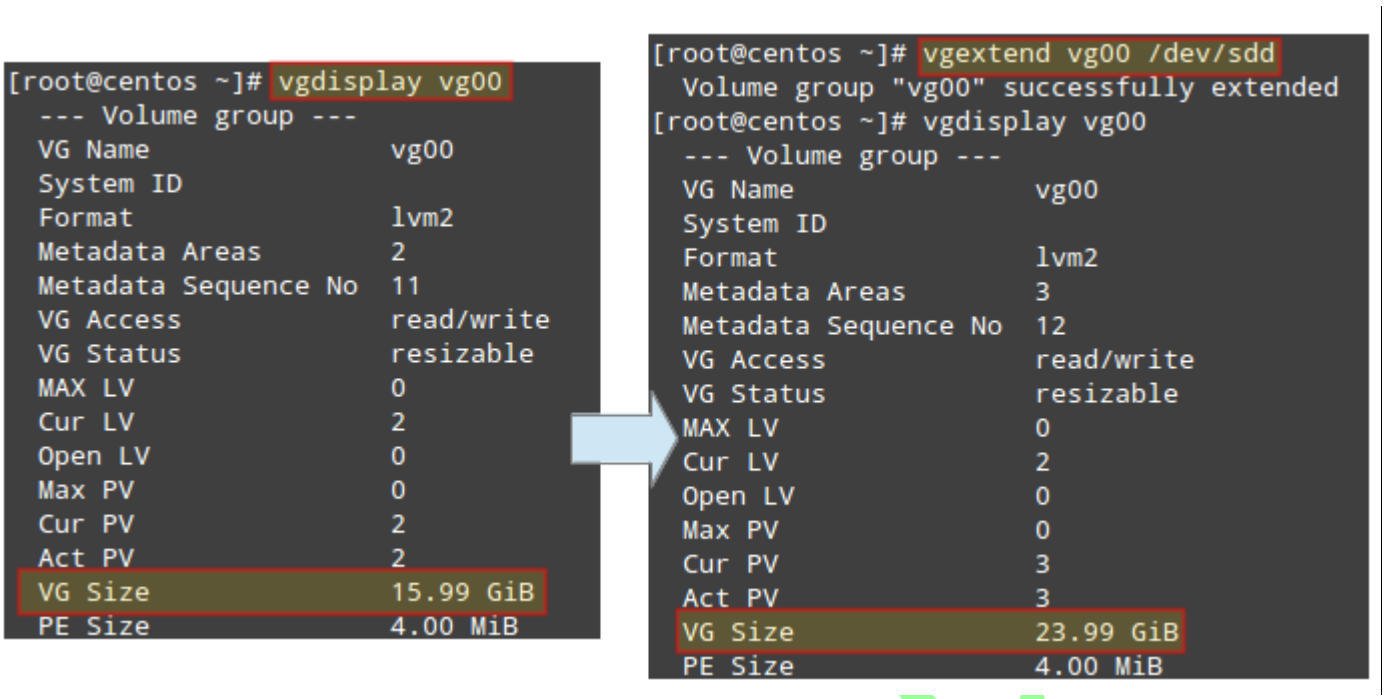
It is important to include the minus (-) or plus (+) signs while resizing a logical volume. Otherwise, you're setting a fixed size for the LV instead of resizing it.

It can happen that you arrive at a point when resizing logical volumes cannot solve your storage needs anymore and you need to buy an extra storage device. Keeping it simple, you will need another disk. We are going to simulate this situation by adding the remaining PV from our initial setup (/dev/sdd).

To add /dev/sdd to vg00, do

```
vgextend vg00 /dev/sdd
```

If you run vgdisplay vg00 before and after the previous command, you will see the increase in the size of the VG:



Now you can use the newly added space to resize the existing LVs according to your needs, or to create additional ones as needed.

Mounting logical volumes on boot and on demand

Of course there would be no point in creating logical volumes if we are not going to actually use them! To better identify a logical volume we will need to find out what its UUID (a non-changing attribute that uniquely identifies a formatted storage device) is. To do that, use blkid followed by the path to each device:

```
blkid /dev/vg00/vol_projects
blkid /dev/vg00/vol_backups
```

```
[root@centos ~]# blkid /dev/vg00/vol_projects
/dev/vg00/vol_projects: UUID="b85df913-580f-461c-844f-546d8cde4646" TYPE="ext4"
[root@centos ~]# blkid /dev/vg00/vol_backups
/dev/vg00/vol_backups: UUID="e1929239-5087-44b1-9396-53e09db6eb9e" TYPE="ext4"
[root@centos ~]#
```

Create mount points for each LV:

```
mkdir /home/projects
mkdir /home/backups
```

and insert the corresponding entries in /etc/fstab (make sure to use the UUIDs obtained before):

```
UUID=b85df913-580f-461c-844f-546d8cde4646 /home/projects        ext4 defaults 0 0
UUID=e1929239-5087-44b1-9396-53e09db6eb9e /home/backups ext4        defaults 0 0
```

Then save the changes and mount the LVs:

```
mount -a
mount | grep home
```

```
[root@centos ~]# mount -a
[root@centos ~]# mount | grep home
/dev/mapper/vg00-vol_projects on /home/projects type ext4 (rw,relatime,seclabel,data=ordered)
/dev/mapper/vg00-vol_backups on /home/backups type ext4 (rw,relatime,seclabel,data=ordered)
[root@centos ~]#
```

When it comes to actually using the LVs, you will need to assign proper ugo+rwx permissions as explained in Chapter 8 (“User management and file attributes”).

Summary

In this article we have introduced Logical Volume Management, a versatile tool to manage storage devices that provides scalability. When combined with RAID, you can enjoy not only scalability (provided by LVM) but also redundancy (offered by RAID). In this type of setup, you will typically find LVM on top of RAID, that is, configure RAID first and then configure LVM on top of it.

Chapter 12: System documentation

Once you get used to working with the command line and feel comfortable doing so, you realize that a regular Linux installation includes all the documentation you need to use and configure the system. Another good reason to become familiar with command line help tools is that in the LFCS and LFCE exams, those are the only sources of information you can use - no internet browsing and no googling. It's just you and the command line.

For that reason, in this article we will give you some tips to effectively use the installed docs and tools in order to prepare to pass the Linux Foundation Certification exams.

Man pages

A man page, short for manual page, is nothing less and nothing more than what the word suggests: a manual for a given tool. It contains the list of options (with explanation) that the command supports, and some man pages even include usage examples as well.

To open a man page, use the **man** command followed by the name of the tool you want to learn more about. For example:

```
man diff
```

will open the manual page for diff, a tool used to compare text files line by line (to exit, simply hit the q key.).

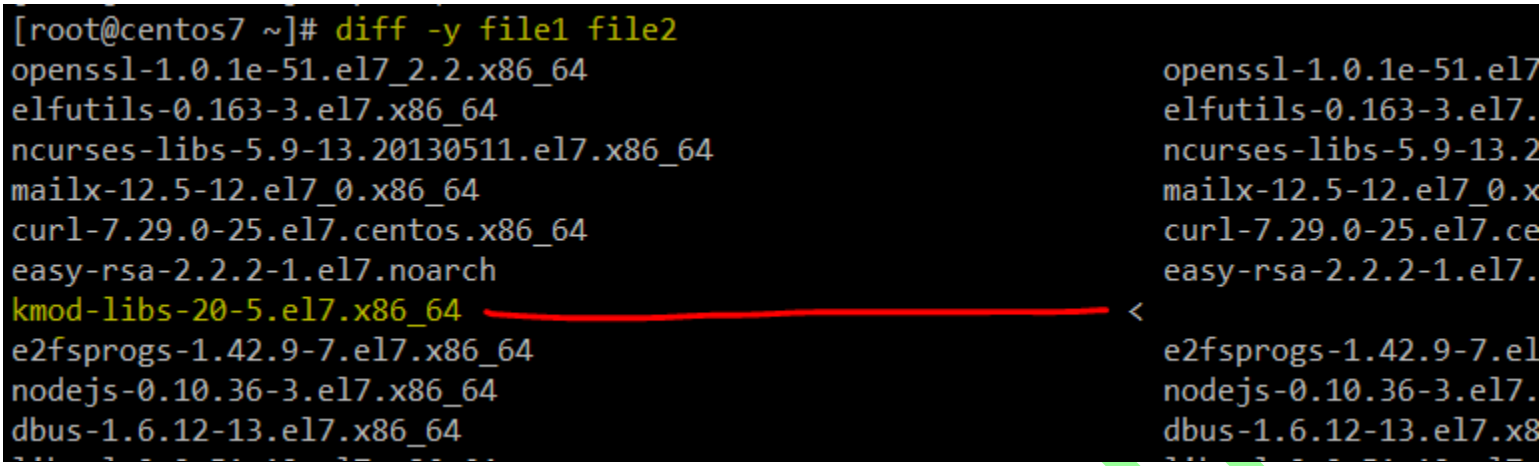
Let's say we want to compare two text files named file1 and file2. These files contain the list of packages that are installed in two Linux boxes with the same distribution and version. Doing a diff between file1 and file2 will tell us if there is a difference between those lists:

```
diff file1 file2
```

```
[root@centos7 ~]# diff file1 file2
7d6
< kmod-libs-20-5.el7.x86_64
24d22
< libunistring-0.9.3-9.el7.x86_64
41d38
< pth-2.0.7-23.el7.x86_64
65,67d61
< libpipeline-1.2.3-3.el7.x86_64
< cpio-2.11-24.el7.x86_64
< perl-Net-HTTP-6.06-2.el7.noarch
[root@centos7 ~]#
```

where the < sign indicates lines missing in file2. If there were lines missing in file1, they would be indicated by the > sign instead. On the other hand, 7d6 means line #7 in file should be deleted in order to match file2 (same with 24d22 and 41d38), and 65,67d61 tells us we need to remove lines 65 through 67 in file one. If we make these corrections, both files will then be identical.

Alternatively, you can display both files side by side using the -y option, according to the man page. You may find this helpful to more easily identify missing lines in files:



Also, you can use diff to compare two binary files. If they are identical, diff will exit silently without output. Otherwise, it will return the following message: “Binary files X and Y differ”.

The --help option

The --help option, available in many (if not all) commands, can be considered a short manual page for that specific command. Although it does not provide a comprehensive description of the tool, it is an easy way to obtain information on the usage of a program and a list of its available options at a quick glance.

For example,

```
sed --help
```

shows the usage of each option available in sed (the stream editor).

One of the classic examples of using sed consists of replacing characters in files. Using the -i option (described as “edit files in place”), you can edit a file without opening it. If you want to make a backup of the original contents as well, use the -i option followed by a SUFFIX to create a separate file with the original contents.

For example, to replace each occurrence of the word Lorem with Tecmint (case insensitive) in lorem.txt and create a new file with the original contents of the file, do:

```
sed -i.orig 's/Lorem/Tecmint/gI' lorem.txt
```

Please note that every occurrence of Lorem has been replaced with Tecmint in lorem.txt, and the original contents of lorem.txt has been saved to lorem.txt.orig.

TIP: You can create your own random Lorem Ipsum text here.

```
[root@centos7 test]# ls
lorem.txt
[root@centos7 test]# less lorem.txt | grep -i lorem
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas vehicula matti
us purus sem, euismod eu lacus sit amet, dictum consequat augue. Aenean sodales,
i. Pellentesque a ex eu felis interdum efficitur. Integer ultricies elit risus,
m tempus orci, in posuere metus quam id turpis. Cras porta erat ut est sollicitu
Sed nibh ex, faucibus a pellentesque at, ornare at nunc. Nullam eu lorem sapien.
 purus, a feugiat lacus libero sed eros. Maecenas at porta sem, sed porttitor ju
Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus
apibus eros, posuere aliquet neque luctus vel. Suspendisse fermentum dapibus dui
ut. Aliquam non auctor ex. Nunc a ornare erat. Nulla a dignissim odio, quis posu
ectus pulvinar eleifend. Duis eu mauris non sem laoreet ultricies ac nec erat. D
Proin et efficitur augue. Cras nec ligula ligula. Fusce id pellentesque velit, v
egestas eleifend libero ut molestie. Integer elementum orci ut turpis lobortis t
ent per conubia nostra, per inceptos himenaeos. Aenean et tempor sem, a viverra
la ut sollicitudin finibus, justo dolor feugiat massa, in rhoncus turpis mi a pu
 eros mollis aliquam. Phasellus ut maximus erat, quis porttitor nibh. Donec faci
aoreet.
[root@centos7 test]# sed -i.orig 's/Lorem/Tecmint/gI' lorem.txt
[root@centos7 test]# ls
lorem.txt  lorem.txt.orig
[root@centos7 test]# less lorem.txt | grep -i lorem
[root@centos7 test]# less lorem.txt.orig | grep -i lorem
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas vehicula matti
us purus sem, euismod eu lacus sit amet, dictum consequat augue. Aenean sodales,
i. Pellentesque a ex eu felis interdum efficitur. Integer ultricies elit risus,
m tempus orci, in posuere metus quam id turpis. Cras porta erat ut est sollicitu
Sed nibh ex, faucibus a pellentesque at, ornare at nunc. Nullam eu lorem sapien.
 purus, a feugiat lacus libero sed eros. Maecenas at porta sem, sed porttitor ju
```

Installed documentation in /usr/share/doc

This is probably my favorite pick. If you go to /usr/share/doc and do a directory listing, you will see lots of directories with the names of the installed tools in your Linux system. According to the Filesystem Hierarchy Standard, these directories contain useful information that might not be in the man pages, along with templates and configuration files to make configuration easier.

For example, let's consider squid-3.3.8 (version may vary from distribution to distribution) for the popular HTTP proxy and cache server.

Let's cd into that directory:

```
cd /usr/share/doc/squid-3.3.8
```

and do a directory listing:

```
[root@centos7 squid-3.3.8]# ls
ChangeLog  COPYRIGHT   README    rredir.pl              url-normalizer.pl
COPYING    QUICKSTART  rredir.c  squid.conf.documented  user-agents.pl
[root@centos7 squid-3.3.8]#
```

You may want to pay special attention to QUICKSTART and squid.conf.documented. These files contain an extensive documentation about Squid and a heavily commented configuration file, respectively. For other packages, the exact names may differ (as QuickRef or 00QUICKSTART, for example), but the principle is the same.

Other packages, such as the Apache web server, provide configuration file templates inside /usr/share/doc, that will be helpful when you have to configure a standalone server or a virtual host, to name a few cases.

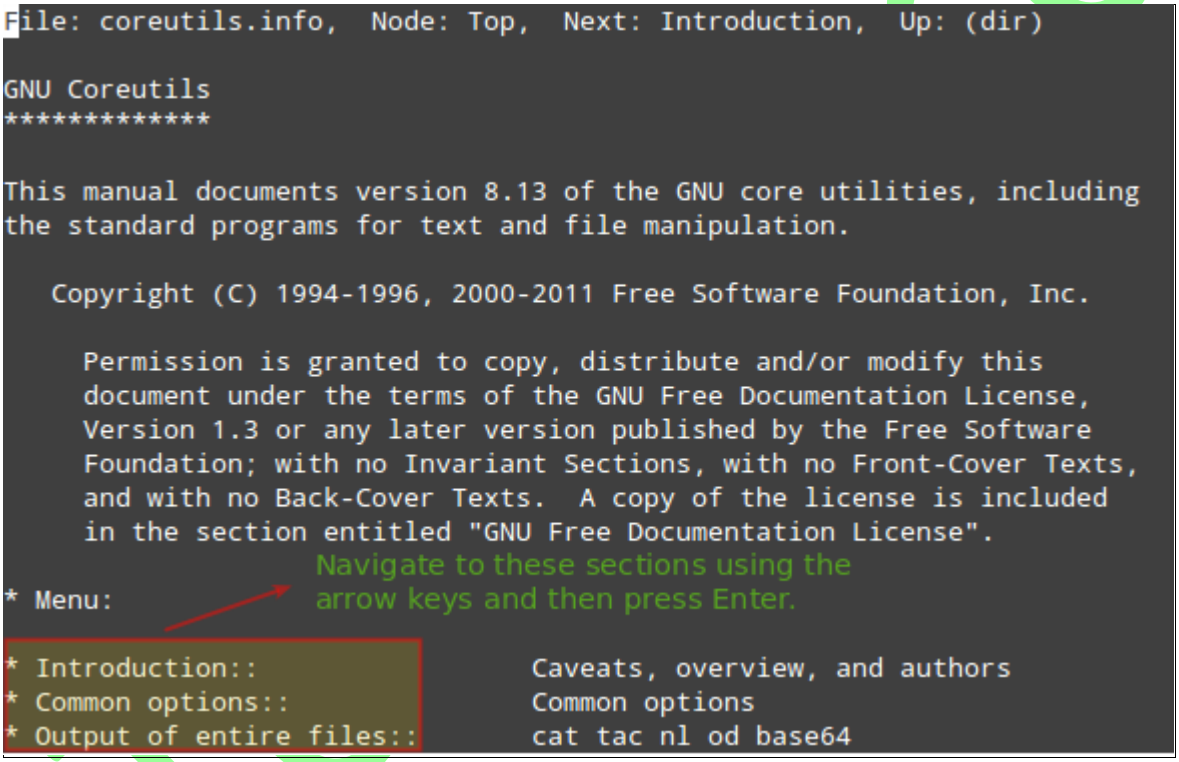
GNU info

You can think of info documents as man pages on steroids. As such, they not only provide help for a specific tool, but also they do so with hyperlinks (yes, hyperlinks in the command line!) that allow you to navigate from a section to another using the arrow keys and Enter to confirm.

Perhaps the most illustrative example is

```
info coreutils
```

Since coreutils contains the basic file, shell and text manipulation utilities which are expected to exist on every operating system, you can reasonably expect a detailed description for each one of those categories in info coreutils.



As it is the case with man pages, you can exit an info document by pressing the q key.

Additionally, GNU info can be used to display regular man pages as well when followed by the tool name. For example:

```
info tune2fs
```

will return the man page of tune2fs, the ext2/3/4 filesystems management tool.

And now that we're at it, let's review some of the uses of tune2fs:

Display information about the filesystem on top of /dev/mapper/vg00-vol_backups:

```
tune2fs -l /dev/mapper/vg00-vol_backups
```

Set a filesystem volume name (Backups in this case):

```
tune2fs -L Backups /dev/mapper/vg00-vol_backups
```

Change the check intervals and / or mount counts (use the -c option to set a number of mount counts and / or the -i option to set a check interval, where d=days, w=weeks, and m=months)

```
tune2fs -c 150 /dev/mapper/vg00-vol_backups # Check every 150 mounts
tune2fs -i 6w /dev/mapper/vg00-vol_backups # Check every 6 weeks
```

All of the above options can be listed with the --help option, or viewed in the man page.

Summary

Regardless of the method that you choose to invoke help for a given tool, knowing that they exist and how to use them will certainly come in handy in the exam. Keep in mind that during the exam you will not be able to use any other source of documentation except what is installed or can be installed as a package using the command line.

Chapter 13: The Grand Unified Bootloader (GRUB)

The Linux boot process –from the time you press the power button of your computer until you get a fully-functional system- follows this high-level sequence (as we discussed in Chapter 7, "Managing the startup process and related services", where we introduced the service management systems and tools used by modern Linux distributions):

1. A process known as POST (Power-On Self Test) performs an overall check on the hardware components of your computer.
2. When POST completes, it passes the control over to the boot loader, which in turn loads the Linux kernel in memory (along with initramfs) and executes it. The most used boot loader in Linux is the GRand Unified Boot loader, or GRUB for short.
3. The kernel checks and accesses the hardware, and then runs the initial process (mostly known by its generic name "init") which in turn completes the system boot by starting services.

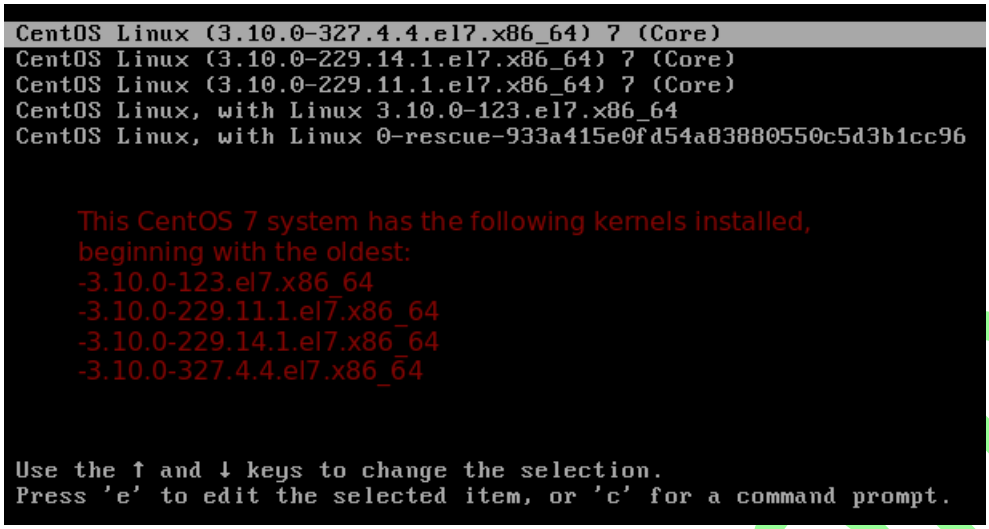
In this article we will introduce you to GRUB and explain why a boot loader is necessary, and how it adds versatility to the system.

Introducing GRUB

Two major GRUB versions (v1 -sometimes called GRUB Legacy- and v2) can be found in modern systems, although most distributions use v2 by default in their latest versions. Only Red Hat Enterprise Linux 6 and its derivatives still use v1 today. Thus, we will focus primarily on the features of v2 in this guide.

Regardless of the GRUB version, a boot loader allows the user to 1) modify the way the system behaves by specifying different kernels to use, 2) choose between alternate operating systems to boot, and 3) add or edit configuration stanzas to change boot options, among other things. Today, GRUB is maintained by the GNU project and is well documented in their website. You are encouraged to use the official documentation while going through this guide.

When the system boots you are presented with the following GRUB screen in the main console. Initially, you are prompted to choose between alternate kernels (by default, the system will boot using the latest kernel) and are allowed to enter a GRUB command line (with c) or edit the boot options (by pressing the e key).



One of the reasons why you would consider booting with an older kernel is a hardware device that used to work properly and has started "acting up" after an upgrade (refer to this link in the AskUbuntu forums for an example).

The GRUB v2 configuration is read on boot from /boot/grub/grub.cfg or /boot/grub2/grub.cfg, whereas /boot/grub/grub.conf or /boot/grub/menu.lst are used in v1. These files are NOT to be edited by hand, but are modified based on the contents of /etc/default/grub and the files found inside /etc/grub.d.

In a CentOS 7, here's the configuration file that is created when the system is first installed:

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="vconsole.keymap=la-latin1 rd.lvm.lv=centos_centos7-2/swap
crashkernel=auto  vconsole.font=latarcyrheb-sun16 rd.lvm.lv=centos_centos7-2/root
rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
```

In addition to the online documentation, you can also find the GNU GRUB manual using info as follows:

```
info grub
```

If you're interested specifically in the options available for /etc/default/grub, you can invoke the configuration section directly:

```
info -f grub -n 'Simple configuration'
```

Using the command above you will find out that GRUB_TIMEOUT sets the time between the moment when the initial screen appears and the system automatic booting begins unless interrupted by the user. When this variable is set to -1, boot will not be started until the user makes a selection.

When multiple operating systems or kernels are installed in the same machine, GRUB_DEFAULT requires an integer value that indicates which OS or kernel entry in the GRUB initial screen should be selected to boot by default. The list of entries can be viewed not only in the splash screen shown above, but also using the following command:

In CentOS and openSUSE:

```
awk -F\' '$1=="menuentry " {print $2}' /boot/grub2/grub.cfg
```

In Ubuntu:

```
awk -F\' '$1=="menuentry " {print $2}' /boot/grub/grub.cfg
```

In the example shown in the below image, if we wish to boot with the kernel version 3.10.0-123.el7.x86_64 (4th entry), we need to set GRUB_DEFAULT to 3 (entries are internally numbered beginning with zero) as follows:

```
GRUB_DEFAULT=3
```

```
[root@server ~]# awk -F\' '$1=="menuentry " {print $2}' /boot/grub2/grub.cfg
CentOS Linux (3.10.0-327.4.4.el7.x86_64) 7 (Core)
CentOS Linux (3.10.0-229.14.1.el7.x86_64) 7 (Core)
CentOS Linux (3.10.0-229.11.1.el7.x86_64) 7 (Core)
CentOS Linux (3.10.0-123.el7.x86_64) 7 (Core)
CentOS Linux (0-rescue-933a415e0fd54a83880550c5d3b1cc96) 7 (Core)
[root@server ~]#
```

One final GRUB configuration variable that is of special interest is GRUB_CMDLINE_LINUX, which is used to pass options to the kernel. The options that can be passed through GRUB to the kernel are well documented in the Kernel Parameters file and in man 7 bootparam.

Current options in my CentOS 7 server are:

```
GRUB_CMDLINE_LINUX="vconsole.keymap=la-latin1 rd.lvm.lv=centos_centos7-2/swap
crashkernel=auto  vconsole.font=latarcyrheb-sun16 rd.lvm.lv=centos_centos7-2/root
rhgb quiet"
```

Why would you want to modify the default kernel parameters or pass extra options? In simple terms, there may be times when you need to tell the kernel certain hardware parameters that it may not be able to determine on its own, or to override the values that it would detect. This happened to me not too long ago when I tried Vector Linux, a derivative of Slackware, on my 10-year old laptop. After installation it did not detect the right settings for my video card so I had to modify the kernel options passed through GRUB in order to make it work.

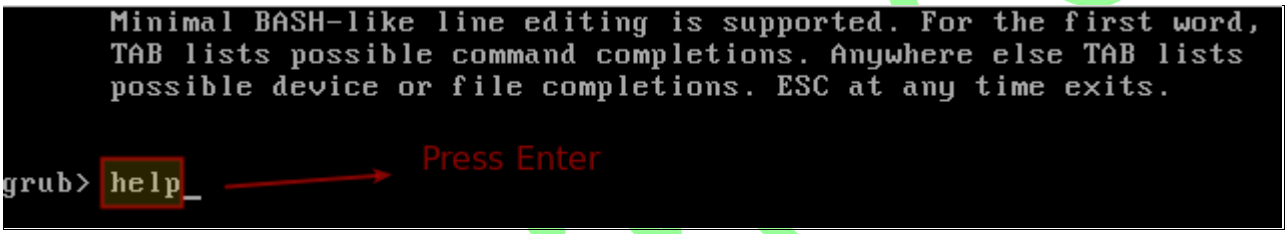
Another example is when you need to bring the system to single-user mode to perform maintenance tasks. You can do this by appending the word single to GRUB_CMDLINE_LINUX and rebooting:

```
GRUB_CMDLINE_LINUX="vconsole.keymap=la-latin1 rd.lvm.lv=centos_centos7-2/swap
crashkernel=auto  vconsole.font=latarcyrheb-sun16 rd.lvm.lv=centos_centos7-2/root
rhgb quiet single"
```

After editing /etc/defalt/grub, you will need to run update-grub (Ubuntu) or grub2-mkconfig -o /boot/grub2/grub.cfg (CentOS and openSUSE) afterwards to update grub.cfg (otherwise, changes will be lost upon boot). This command will process the boot configuration files mentioned earlier to update grub.cfg. This method ensures changes are permanent, while options passed through GRUB at boot time will only last during the current session.

Fixing GRUB issues

If you install a second operating system or if your GRUB configuration file gets corrupted due to human error, there are ways you can get your system back on its feet and be able to boot again. In the initial screen, press c to get a GRUB command line (remember that you can also press e to edit the default boot options), and use help to bring the available commands in the GRUB prompt:

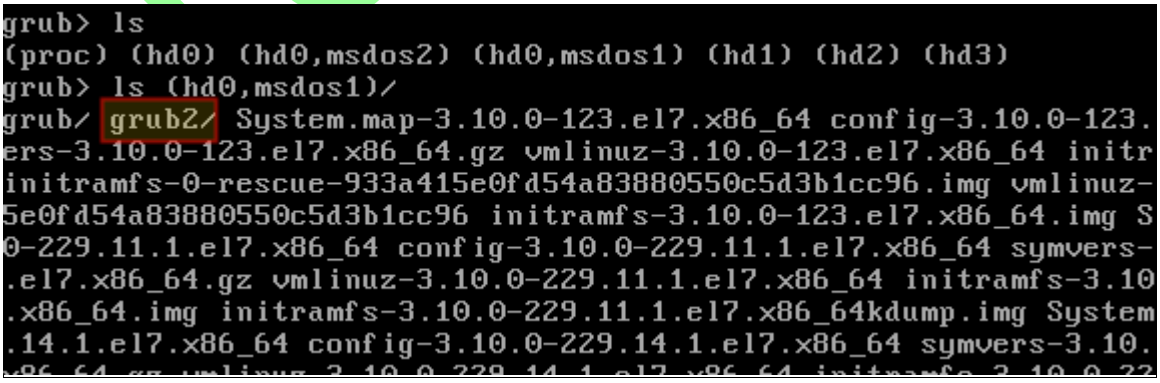


We will focus on ls, which will list the installed devices and filesystems, and we will examine what it finds. In the image below we can see that there are 4 hard drives (hd0 through hd3). Only hd0 seems to have been partitioned (as evidenced by msdos1 and msdos2, where 1 and 2 are the partition numbers and msdos is the partitioning scheme).

Let's now examine the first partition on hd0 (msdos1) to see if we can find GRUB there. This approach will allow us to boot Linux and there use other high level tools to repair the configuration file or reinstall GRUB altogether if it is needed:

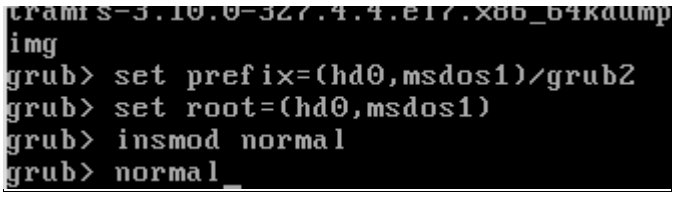
```
ls (hd0,msdos1)/
```

As we can see in the highlighted area, we found the grub2 directory in this partition:



Once we are sure that GRUB resides in (hd0,msdos1), let's tell GRUB where to find its configuration file and then instruct it to attempt to launch its menu:

```
set prefix=(hd0,msdos1)/grub2
set root=(hd0,msdos1)
insmod normal
normal
```



Then in the GRUB menu, choose an entry and press Enter to boot using it. Once the system has booted you can issue the grub2-install /dev/sdX command (change sdX with the device you want to install GRUB on). The boot information will then be updated and all related files be restored.

Other more complex scenarios are documented, along with their suggested fixes, in the Ubuntu GRUB2 Troubleshooting guide. The concepts explained there are valid for other distributions as well.

Summary

In this article we have introduced you to GRUB, indicated where you can find documentation -both online and offline-, and explained how to approach an scenario where a system has stopped booting properly due to a bootloader-related issue. Fortunately, GRUB is one of the tools that is best documented and you can easily find help either in the installed docs or online using the resources we have shared in this chapter

Chapter 14: Integrity and availability

Every Linux system administrator needs to know how to verify the integrity and availability of hardware, resources, and key processes. In addition, setting resource limits on a per-user basis must also be a part of his / her skill set. In this article we will explore a few ways to ensure that the system -both hardware and the software- is behaving correctly to avoid potential issues that may cause unexpected production downtime and money loss.

Reporting processors statistics

With mpstat you can view the activities for each processor individually or the system as a whole, both as a one-time snapshot or dynamically. In order to use this tool, you will need to install **sysstat**:

In CentOS:

```
yum update && yum install sysstat
```

In Ubuntu:

```
aptitutde update && aptitude install sysstat
```

In openSUSE:

```
zypper update && zypper install sysstat
```

Once you have installed this tool, you can use it to generate reports of processors statistics.

To display 3 global reports of CPU utilization (-u) for all CPUs (as indicated by -P ALL) at a 2-second interval, do:

```
mpstat -P ALL -u 2 3
```

To view the same statistics for a specific CPU (CPU 0 in the following example), use

```
mpstat -P 0 -u 2 3
```

The output of the above commands shows these columns:

- CPU: Processor number as an integer, or the word all as an average for all processors.
- %usr: Percentage of CPU utilization while running user level applications.
- %nice: Same as %usr, but with nice priority.
- %sys: Percentage of CPU utilization that occurred while executing kernel applications. This does not include time spent dealing with interrupts or handling hardware..
- %iowait: Percentage of time when the given CPU (or all) was idle, during which there was a resource-intensive I/O operation scheduled on that CPU. A more detailed explanation (with examples) can be found in http://veithen.github.io/2013/11/18/iowait-linux.html.
- %irq: Percentage of time spent servicing hardware interrupts.

- %soft: Same as %irq, but with software interrupts.
- %steal: Percentage of time spent in involuntary wait (steal or stolen time) when a virtual machine, as guest, is "winning" the hypervisor's attention while competing for the CPU(s). This value should be kept as small as possible. A high value in this field means the virtual machine is stalling - or soon will be.
- %guest: Percentage of time spent running a virtual processor.
- %idle: percentage of time when CPU(s) were not executing any tasks. If you observe a low value in this column, that is an indication of the system being placed under a heavy load. In that case, you will need to take a closer look at the process list, as we will discuss in a minute, to determine what is causing it.

To put the place the processor under a somewhat high load, run the following commands and then execute mpstat (as indicated) in a separate terminal:

```
dd if=/dev/zero of=test.iso bs=1G count=1
mpstat -u -P 0 2 3
ping -f localhost # Interrupt with Ctrl + C after mpstat below completes
mpstat -u -P 0 2 3
```

Finally, compare to the output of mpstat under "normal" circumstances:

```
[root@server ~]# mpstat -u -P 0 2 3
Linux 3.10.0-327.4.4.el7.x86_64 (server)        03/23/2016      _x86_64_        (1 CPU)
Statistics while dd was creating the empty 1 GB file:
10:51:24 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
10:51:26 PM    0    0.00    0.00   22.78   70.89    0.00    6.33    0.00    0.00    0.00    0.00
10:51:28 PM    0    0.65    0.00   27.10   68.39    0.00    3.87    0.00    0.00    0.00    0.00
10:51:30 PM    0    0.00    0.00   28.30   68.55    0.00    3.14    0.00    0.00    0.00    0.00
Average:       0    0.21    0.00   26.06   69.28    0.00    4.45    0.00    0.00    0.00    0.00
[root@server ~]# mpstat -u -P 0 2 3
Linux 3.10.0-327.4.4.el7.x86_64 (server)        03/23/2016      _x86_64_        (1 CPU)
Statistics while the flood ping (-f) was being performed:
10:53:12 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
10:53:14 PM    0    0.00    0.00  100.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
10:53:16 PM    0    0.00    0.00   99.00    0.00    0.00    1.00    0.00    0.00    0.00    0.00
10:53:18 PM    0    0.00    0.00   99.50    0.00    0.00    0.50    0.00    0.00    0.00    0.00
Average:       0    0.00    0.00   99.50    0.00    0.00    0.50    0.00    0.00    0.00    0.00
[root@server ~]# mpstat -u -P 0 2 3
Linux 3.10.0-327.4.4.el7.x86_64 (server)        03/23/2016      _x86_64_        (1 CPU)
Statistics under "normal" circumstances:
10:53:33 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
10:53:35 PM    0    0.00    0.00    0.00    0.00    0.00    0.50    0.00    0.00    0.00   99.50
10:53:37 PM    0    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
10:53:39 PM    0    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  100.00
Average:       0    0.00    0.00    0.00    0.00    0.00    0.17    0.00    0.00    0.00   99.83
```

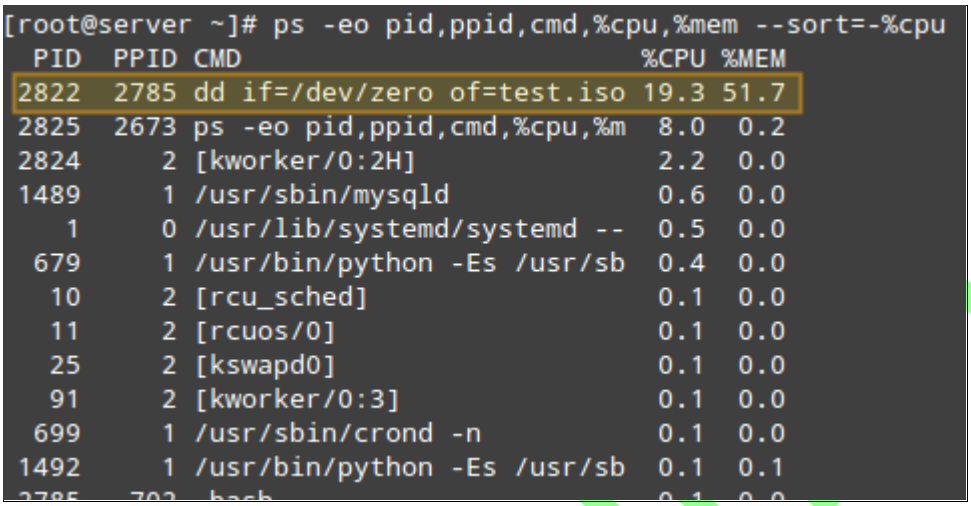
As you can see in the image above, CPU 0 was under a heavy load during the first two examples, as indicated by the %idle column. In the next section we will discuss how to identify these resource-hungry processes, how to obtain more information about them, and how to take appropriate action.

Reporting processes

To list processes sorting them by CPU usage, we will use the well-known **ps** command with the **-eo** (to select all processes with user-defined format) and **--sort** (to specify a custom sorting order) options, like so:

```
ps -eo pid,ppid,cmd,%cpu,%mem --sort=-%cpu
```

The above command will only show the PID, PPID, the command associated with the process, and the percentage of CPU and RAM usage sorted by the percentage of CPU usage in descending order. When executed during the creation of the .iso file, here's the first few lines of the output:



Once we have identified a process of interest (such as the one with PID=2822), we can navigate to /proc/PID (/proc/2822 in this case) and do a directory listing. This directory is where several files and subdirectories with detailed information about this particular process are kept while it is running.

For example:

1) /proc/2822/io contains IO statistics for the process (number of characters and bytes read and written, among others, during IO operations).

2) /proc/2822/attr/current shows the current SELinux security attributes of the process.

3) /proc/2822/cgroup describes the control groups (cgroups for short) to which the process belongs if the CONFIG_CGROUPS kernel configuration option is enabled, which you can verify with

cat /boot/config-$(uname -r) | grep -i cgroups

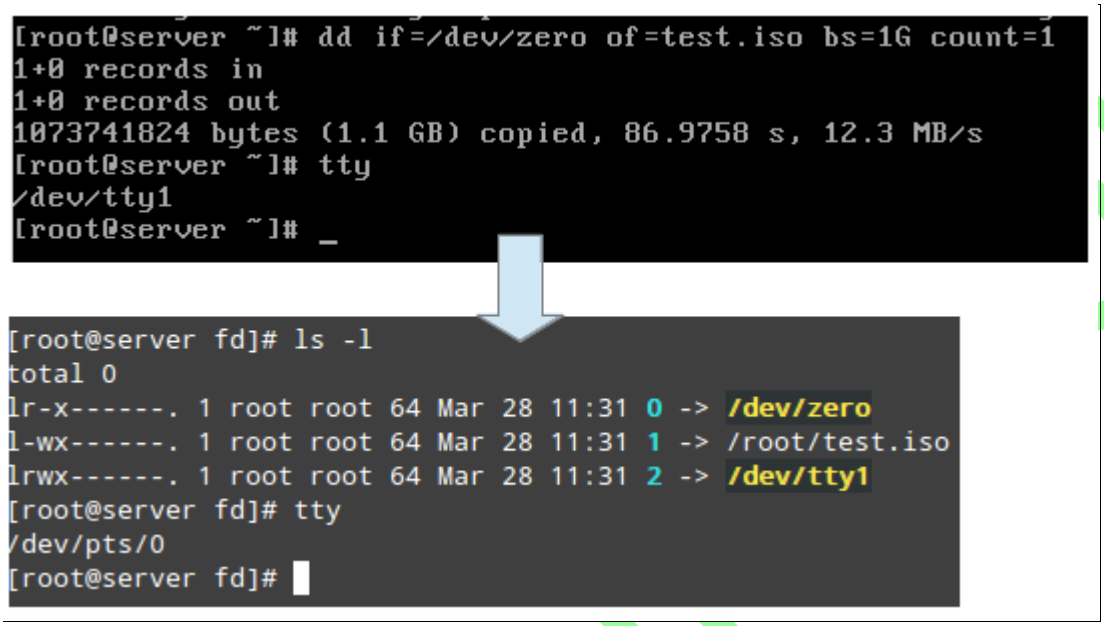
If the option is enabled, you should see

```
CONFIG_CGROUPS=y
```

in the output.

Using cgroups you can manage the amount of allowed resource usage on a per-process basis as explained in Chapters 1 through 4 of the Red Hat Enterprise Linux 7 Resource Management guide, in Chapter 9 of the openSUSE System Analysis and Tuning guide, and in the Control Groups section of the Ubuntu 14.04 Server documentation.

4) /proc/2822/fd is a directory that contains one symbolic link for each file descriptor the process has opened. The following image shows this information for the process that was started in tty1 (the first terminal) to create the .iso image:



The above image shows that stdin (file descriptor 0), stdout (file descriptor 1), and stderr (file descriptor 2) are mapped to /dev/zero, /root/test.iso, and /dev/tty1, respectively.

More information about /proc can be found in "The /proc filesystem" document kept and maintained by Kernel.org, and in the Linux Programmer's Manual.

Setting resource limits on a per-user basis

If you are not careful and allow any user to run an unlimited number of processes, you may eventually experience an unexpected system shutdown or get locked out as the system enters an unusable state. To prevent this from happening, you should place a limit on the number of processes users can start.

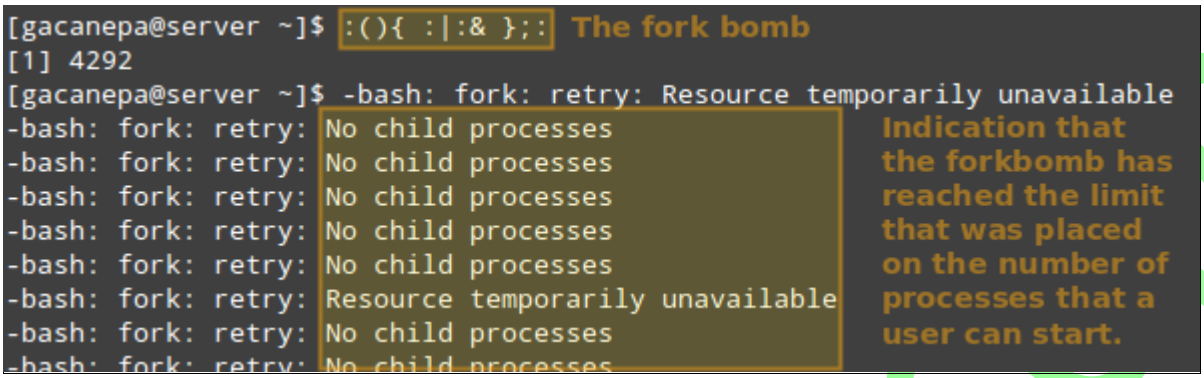
To do this, edit /etc/security/limits.conf and add the following line at the bottom of the file to set the limit:

```
*          hard        nproc   10
```

The first field can be used to indicate either a user, a group, or all of them (*), whereas the second field enforces a hard limit on the number of process (nproc) to 10. To apply changes, logging out and back in is enough.

Thus, let's see what happens if a certain user other than root (either a legitimate one or not) attempts to start a shell fork bomb. If we had not implemented limits, this would initially launch two instances of a function, and then duplicate each of them in a neverending loop. Thus, it would eventually bringing your system to a crawl.

However, with the above restriction in place, the fork bomb does not succeed but the user will still get locked out until the system administrator kills the process associated with it:



TIP: Other possible restrictions made possible by ulimit are documented in the limits.conf file.

Other process management tools

In addition to the tools discussed previously, a system administrator may also need to:

a) Modify the execution priority (use of system resources) of a process using renice. This means that the kernel will allocate more or less system resources to the process based on the assigned priority (a number commonly known as "niceness" in a range from -20 to 19). The lower the value, the greater the execution priority. Regular users (other than root) can only modify the niceness of processes they own to a higher value (meaning a lower execution priority), whereas root can modify this value for any process, and may increase or decrease it.

The basic syntax of renice is as follows:

```
renice [-n] <new priority> <UID, GID, PGID, or empty> identifier
```

If the argument after the new priority value is not present (empty), it is set to PID by default. In that case, the niceness of process with PID=identifier is set to <new priority>.

b) Interrupt the normal execution of a process when needed. This is commonly known as "killing" the process. Under the hood, this means sending the process a signal to finish its execution properly and release any used resources in an orderly manner.

To kill a process, use the kill command as follows:

```
kill PID
```

Alternatively, you can use pkill to terminate all processes of a given owner (-u), or a group owner (-G), or even those processes which have a PPID in common (-P). These options may be followed by the numeric representation or the actual name as identifier:

```
pkill [options] identifier
```

For example,

```
pkill -G 1000
```

will kill all processes owned by group with GID=1000

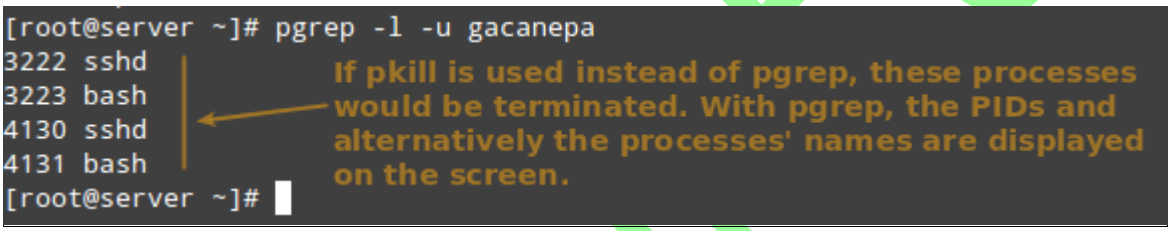
and

```
pkill -P 4993
```

will kill all processes whose PPID is 4993.

Before running a pkill, it is a good idea to test the results with pgrep first, perhaps using the -l option as well to list the processes' names. It takes the same options but only returns the PIDs of processes (without taking any further action) that would be killed if pkill is used.

This is illustrated in the next image:



Summary

In this article we have explored a few ways to monitor resource usage in order to verify the integrity and availability of critical hardware and software components in a Linux system. We have also learned how to take appropriate action (either by adjusting the execution priority of a given process or by terminating it) under unusual circumstances.

Chapter 15: Kernel runtime parameters

In Chapter 13 ("GRUB and the boot process") we explained how to use GRUB to modify the behavior of the system by passing options to the kernel for the ongoing boot process. Similarly, you can use the command line in a running Linux system to alter certain runtime kernel parameters as a one-time modification, or permanently by editing a configuration file. Thus, you are allowed to enable or disable kernel parameters on-the-fly without much difficulty when it is needed due to a required change in the way the system is expected to operate.

Introducing the /proc filesystem

The latest specification of the Filesystem Hierarchy Standard indicates that /proc represents the default method for handling process and system information as well as other kernel and memory information. Particularly, /proc/sys is where you can find all the information about devices, drivers, and some kernel features.

The actual internal structure of /proc/sys depends heavily on the kernel being used, but you are likely to find the following directories inside. In turn, each of them will contain other subdirectories where the values for each parameter category are maintained:

- **dev**: parameters for specific devices connected to the machine.
- **fs**: filesystem configuration (quotas and inodes, for example).
- **kernel**: kernel-specific configuration.
- **net**: network configuration.
- **vm**: use of the kernel's virtual memory.

To modify the kernel runtime parameters we will use the sysctl command. The exact number of parameters that can be modified can be viewed with

```
sysctl -a | wc -l
```

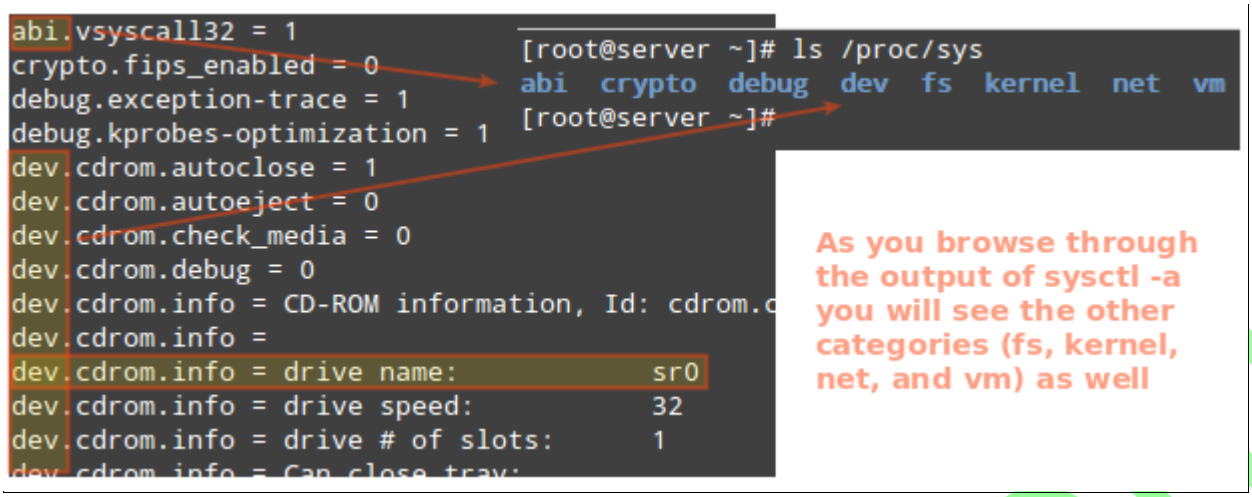
If you want to view the complete list of parameters, just do

```
sysctl -a
```

As the the output of the above command will consist of A LOT of lines, we can use a pipeline followed by less to inspect it more carefully:

```
sysctl -a | less
```

Let's take a look at the first few lines. Please note that the first characters in each line match the names of the directories inside /proc/sys:



For example, the highlighted line

```
dev.cdrom.info = drive name:                    sr0
```

indicates that sr0 is an alias for the optical drive. In other words, that is how the kernel "sees" that drive and uses that name to refer to it. In the following section we will explain how to change other "more important" kernel runtime parameters.

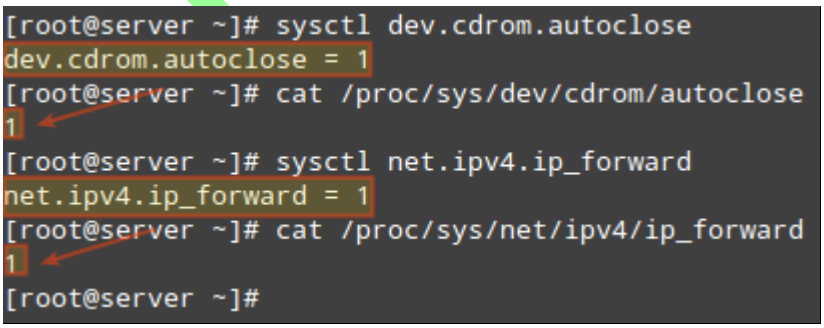
Other kernel runtime parameters explained

Based on what we have explained so far, it is easy to see that the name of a parameter matches the directory structure inside /proc/sys where it can be found. For example:

```
dev.cdrom.autoclose → /proc/sys/dev/cdrom/autoclose
net.ipv4.ip_forward → /proc/sys/net/ipv4/ip_forward
```

That said, we can view the value of a particular parameter using either sysctl followed by the name of the parameter or reading the associated file:

```
sysctl dev.cdrom.autoclose
cat /proc/sys/dev/cdrom/autoclose
sysctl net.ipv4.ip_forward
cat /proc/sys/net/ipv4/ip_forward
```



To set the value for a parameter we can also use sysctl, but using the -w option and followed by the parameter's name, the equal sign, and the desired value. Another method consists of using echo to overwrite the file associated with the parameter. In other words, the following methods are equivalent to disable the packet forwarding functionality in our system (which, by the way, should be the default value when a box is not supposed to pass traffic between networks):

```
echo 0 > /proc/sys/net/ipv4/ip_forward
sysctl -w net.ipv4.ip_forward=0
```

It is important to note that kernel parameters that are set using sysctl will only be enforced during the current session and will disappear when the system is rebooted. To set these values permanently, edit /etc/sysctl.conf with the desired values. For example, to disable packet forwarding in /etc/sysctl.conf make sure this line appears in the file:

```
net.ipv4.ip_forward=0
```

Then run

```
sysctl -p
```

to apply the changes to the running configuration.

Other examples of important kernel runtime parameters are:

- **fs.file-max** specifies the maximum number of file handles the kernel can allocate for the system. Depending on the intended use of your system (web / database / file server, to name a few examples), you may want to change this value to meet the system's needs. Otherwise, you will receive a "Too many open files" error message at best, and may prevent the operating system to boot at the worst. If due to an innocent mistake you find yourself in this last situation, boot in single user mode (as explained in Part 13) and edit /etc/sysctl.conf as instructed earlier. To set the same restriction on a per-user basis, refer to Part 14 of this series.
- **kernel.sysrq** is used to enable the SysRq key in your keyboard (also known as the print screen key) so as to allow certain key combinations to invoke emergency actions when the system has become unresponsive. The default value (16) indicates that the system will honor the Alt+SysRq+key combination and perform the actions listed in the sysrq.c documentation found in kernel.org (where key is one letter in the b-z range). For example, Alt+SysRq+b will reboot the system forcefully (use this as a last resort if your server is unresponsive).

Warning! Do not attempt to press this key combination on a virtual machine because it may force your host system to reboot!

- When set to 1, **net.ipv4.icmp_echo_ignore_all** will ignore ping requests and drop them at the kernel level. This is shown in the below image - note how ping requests are lost after setting this kernel parameter:

```
[root@server ~]# ping -c 2 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.091 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.124 ms

--- localhost ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.091/0.107/0.124/0.019 ms
[root@server ~]# sysctl -w net.ipv4.icmp_echo_ignore_all=1
net.ipv4.icmp_echo_ignore_all = 1
[root@server ~]# ping -c 2 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.

--- localhost ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1000ms

[root@server ~]#
```

A better and easier way to set individual runtime parameters is using .conf files inside /etc/sysctl.d, grouping them by categories. For example, instead of setting net.ipv4.ip_forward=0 and net.ipv4.icmp_echo_ignore_all=1 in /etc/sysctl.conf, we can create a new file named net.conf inside /etc/sysctl.d:

```
echo "net.ipv4.ip_forward=0" > /etc/sysctl.d/net.conf
echo "net.ipv4.icmp_echo_ignore_all=1" >> /etc/sysctl.d/net.conf
```

If you choose to use this approach, do not forget to remove those same lines from /etc/sysctl.conf.

Summary

In this article we have explained how to modify kernel runtime parameters, both persistent and non-persistently, using sysctl, /etc/sysctl.conf, and files inside /etc/sysctl.d. In the sysctl docs you can find more information on the meaning of more variables. Those files represent the most complete source of documentation about the parameters that can be set via **sysctl**.

Chapter 16: SELinux and AppArmor

To overcome the limitations of and to increase the security mechanisms provided by standard ugo/rwx permissions and access control lists, the United States National Security Agency (NSA) devised a flexible Mandatory Access Control (MAC) method known as SELinux (short for Security Enhanced Linux) in order to restrict -among other things- the ability of processes to access or perform other operations on system objects (such as files, directories, network ports, etc) to the least permission possible, while still allowing for later modifications to this model.

Another popular and widely-used MAC is AppArmor, which in addition to the features provided by SELinux, includes a learning mode that allows the system to "learn" how a specific application behaves, and to set limits by configuring profiles for safe application usage.

In CentOS 7, SELinux is incorporated into the kernel itself and is enabled in Enforcing mode by default (more on this in the next section), as opposed to openSUSE and Ubuntu which use AppArmor. In this article we will explain the essentials of SELinux and AppArmor and how to use one of these tools for your benefit depending on your chosen distribution.

SELinux

Security Enhanced Linux can operate in two different ways:

- Enforcing: SELinux denies access based on SELinux policy rules, a set of guidelines that control the security engine.
- Permissive: SELinux does not deny access, but denials are logged for actions that would have been denied if running in enforcing mode.

SELinux can also be disabled. Although it is not an operation mode itself, it is still an option. However, learning how to use this tool is better than just ignoring it. Keep it in mind!

To display the current mode of SELinux, use getenforce. If you want to toggle the operation mode, use setenforce 0 (to set it to Permissive) or setenforce 1 (Enforcing). Since this change will not survive a reboot, you will need to edit the /etc/selinux/config file and set the SELINUX variable to either enforcing, permissive, or disabled in order to achieve persistence across reboots:

```
[root@server ~]# cat /etc/redhat-release
CentOS Linux release 7.2.1511 (Core)
[root@server ~]# getenforce
Enforcing
[root@server ~]# setenforce 0
[root@server ~]# getenforce
Permissive
[root@server ~]# setenforce 1
[root@server ~]# getenforce
Enforcing
[root@server ~]#
```

```
[root@server ~]# cat /etc/selinux/config

# This file controls the state of SELinux on the
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enf
#     permissive - SELinux prints warnings inste
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protecte
```

On a side note, if getenforce returns Disabled, you will have to edit /etc/selinux/config with the desired operation mode and reboot. Otherwise, you will not be able to set (or toggle) the operation mode with setenforce. One of the typical uses of setenforce consists of toggling between SELinux modes (from enforcing to permissive or the other way around) to troubleshoot an application that is misbehaving or not working as expected. If it works after you set SELinux to Permissive mode, you can be confident you're looking at a SELinux permissions issue.

Two classic cases where we will most likely have to deal with SELinux are:

1. Changing the default port where a daemon listens on.
2. Setting the DocumentRoot directive for a virtual host outside of /var/www/html.

Let's take a look at these two cases using the following examples.

EXAMPLE 1: Changing the default port for the sshd daemon

One of the first thing most system administrators do in order to secure their servers is change the port where the SSH daemon listens on, mostly to discourage port scanners and external attackers. To do this, we use the Port directive in /etc/ssh/sshd_config followed by the new port number as follows (we will use port 9999 in this case):

```
Port 9999
```

After attempting to restart the service and checking its status we will see that it failed to start:

```
[root@server ~]# systemctl restart sshd
[root@server ~]# systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabl
   Active: activating (auto-restart) (Result: exit-code) since Mon 2016-06-06 19:10:12
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 10828 ExecStart=/usr/sbin/sshd -D $OPTIONS (code=exited, status=255)
 Main PID: 10828 (code=exited, status=255)

Jun 06 19:10:12 server systemd[1]: sshd.service: main process exited, code=exited, sta
Jun 06 19:10:12 server systemd[1]: Unit sshd.service entered failed state.
Jun 06 19:10:12 server systemd[1]: sshd.service failed.
[root@server ~]#
```

If we take a look at /var/log/audit/audit.log, we will see that sshd was prevented from starting on port 9999 by SELinux because that is a reserved port for the JBoss Management service (SELinux log messages include the word "AVC" so that they might be easily identified from other messages):

```
cat /var/log/audit/audit.log | grep AVC | tail -1
```

```
[root@server ~]# cat /var/log/audit/audit.log | grep AVC | tail -1
type=AVC msg=audit(1465254907.900:6832): avc:  denied  { name_bind } for  pid=16
em_u:object_r:jboss_management_port_t:s0 tclass=tcp_socket
[root@server ~]#
```

At this point most people would probably disable SELinux but we won't. We will see that there's a way for SELinux, and sshd listening on a different port, to live in harmony together. Make sure you have the policycoreutils-python package installed and run

```
semanage port -l | grep ssh
```

to view a list of the ports where SELinux allows sshd to listen on. In the following image we can also see that port 9999 was reserved for another service and thus we can't use it to run another service for the time being:

```
semanage port -l | grep ssh
```

Of course we could choose another port for SSH, but if we are certain that we will not need to use this specific machine for any JBoss-related services, we can then modify the existing SELinux rule and assign that port to SSH instead:

```
semanage port -m -t ssh_port_t -p tcp 9999
```

After that, we can use the first semanage command to check if the port was correctly assigned, or the -lC options (short for list custom):

```
semanage port -lC
semanage port -l | grep ssh
```

```
[root@server ~]# semanage port -l | grep ssh
ssh_port_t                     tcp      22
[root@server ~]# semanage port -l | grep 9999
jboss_management_port_t        tcp      4712, 4447, 7600, 9123, 9990, 9999, 18001
[root@server ~]# semanage port -m -t ssh_port_t -p tcp 9999
[root@server ~]# semanage port -lC
SELinux Port Type              Proto    Port Number

ssh_port_t                     tcp      9999
[root@server ~]# semanage port -l | grep ssh
ssh_port_t                     tcp      9999, 22
[root@server ~]#
```

We can now restart SSH and connect to the service using port 9999. Note that this change WILL survive a reboot.

EXAMPLE 2: Choosing a DocumentRoot outside /var/www/html for a virtual host

If you need to set up a virtual host using a directory other than /var/www/html as DocumentRoot (say, for example, /websrv/sites/gabriel/public_html):

```
DocumentRoot "/websrv/sites/gabriel/public_html"
```

Apache will refuse to serve the content because the index.html has been labeled with the default_t SELinux type, which Apache can't access:

```
[root@server ~]# wget http://localhost/index.html
--2016-06-06 21:14:40--  http://localhost/index.html
Resolving localhost (localhost)... ::1, 127.0.0.1
Connecting to localhost (localhost)|::1|:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2016-06-06 21:14:40 ERROR 403: Forbidden.

[root@server ~]# ls -lZ /websrv/sites/gabriel/public_html/index.html
-rw-r--r--. root root unconfined_u:object_r:default_t:s0 /websrv/sites/gabriel
/public_html/index.html
[root@server ~]#
```

As with the previous example, you can use the following command to verify that this is indeed a SELinux-related issue:

```
cat /var/log/audit/audit.log | grep AVC | tail -1
```

```
[root@server ~]# cat /var/log/audit/audit.log | grep AVC | tail -1
type=AVC msg=audit(1465262080.218:1726): avc:  denied  { getattr } for  pid=48
37 comm="httpd" path="/websrv/sites/gabriel/public_html/index.html" dev="dm-1"
 ino=327257 scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object
_r:default_t:s0 tclass=file
```

To change the label of /websrv/sites/gabriel/public_html recursively to httpd_sys_content_t, do:

```
semanage fcontext -a -t httpd_sys_content_t
"/websrv/sites/gabriel/public_html(/.*)?"
```

The above command will grant Apache read-only access to that directory and its contents.

Finally, to apply the policy (and make the label change effective immediately), do:

```
restorecon -R -v /websrv/sites/gabriel/public_html
```

Now you should be able to access the directory:

```
[root@server ~]# wget http://localhost/index.html
--2016-06-06 21:26:31--  http://localhost/index.html
Resolving localhost (localhost)... ::1, 127.0.0.1
Connecting to localhost (localhost)|::1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1058 (1.0K) [text/html]
Saving to: 'index.html'

100%[======================================>] 1,058       --.-K/s   in 0s

2016-06-06 21:26:31 (101 MB/s) - 'index.html' saved [1058/1058]

[root@server ~]#
```

For more information on SELinux, refer to the Fedora 22 SELinux and Administrator guide.

AppArmor

The operation of AppArmor is based on profiles defined in plain text files where the allowed permissions and access control rules are set. Profiles are then used to place limits on how applications interact with processes and files in the system. A set of profiles is provided out-of-the-box with the operating system, whereas others can be put in place either automatically by applications when they are installed or manually by the system administrator.

Like SELinux, AppArmor runs profiles in two modes. In enforce mode, applications are given the minimum permissions that are necessary for them to run, whereas in complain mode AppArmor allows an application to take restricted actions and saves the "complaints" resulting from that operation to a log (/var/log/kern.log, /var/log/audit/audit.log, and other logs inside /var/log/apparmor). These logs will show -through lines with the word audit in them- errors that would occur should the profile be run in enforce mode. Thus, you can try out an application in complain mode and adjust its behavior before running it under AppArmor in enforce mode.

The current status of AppArmor can be shown using

```
sudo apparmor_status
```

```
gacanepa@ubuntu:~$ sudo apparmor_status
[sudo] password for gacanepa:
apparmor module is loaded.
5 profiles are loaded.
5 profiles are in enforce mode.
   /sbin/dhclient
   /usr/lib/NetworkManager/nm-dhcp-client.action
   /usr/lib/connman/scripts/dhclient-script
   /usr/sbin/mysqld
   /usr/sbin/tcpdump
0 profiles are in complain mode.
1 processes have profiles defined.
1 processes are in enforce mode.
   /usr/sbin/mysqld (1070)
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
gacanepa@ubuntu:~$
```

The image above indicates that the profiles /sbin/dhclient, /usr/sbin/, and /usr/sbin/tcpdump are in enforce mode (that is true by default in Ubuntu). Since not all applications include the associated AppArmor profiles, the apparmor-profiles package, which provides other profiles that have not been shipped by the packages they provide confinement for. By default, they are configured to run in complain mode so that system administrators can test them and choose which ones are desired. We will make use of apparmor-profiles since writing our own profiles is out of the scope of the LFCS certification.

AppArmor profiles are stored inside /etc/apparmor.d. Let's take a look at the contents of that directory before and after installing apparmor-profiles:

```
gacanepa@ubuntu:~$ ls /etc/apparmor.d
abstractions  force-complain  tunables          usr.sbin.tcpdump
cache         local           usr.sbin.mysqld
disable       sbin.dhclient   usr.sbin.rsyslogd   BEFORE
gacanepa@ubuntu:~$
```

```
gacanepa@ubuntu:~$ ls /etc/apparmor.d
abstractions                  usr.lib.dovecot.imap
apache2.d                     usr.lib.dovecot.imap-login
bin.ping                      usr.lib.dovecot.lmtp
cache                         usr.lib.dovecot.log
disable                       usr.lib.dovecot.managesieve
force-complain                usr.lib.dovecot.managesieve-login
local                         usr.lib.dovecot.pop3
program-chunks                usr.lib.dovecot.pop3-login
sbin.dhclient                 usr.lib.dovecot.ssl-params
sbin.klogd                    usr.sbin.avahi-daemon
sbin.syslogd                  usr.sbin.dnsmasq
sbin.syslog-ng                usr.sbin.dovecot
tunables                      usr.sbin.identd
usr.bin.chromium-browser      usr.sbin.mdnsd      AFTER
usr.lib.dovecot.anvil         usr.sbin.mysqld
usr.lib.dovecot.auth          usr.sbin.nmbd
usr.lib.dovecot.config        usr.sbin.nscd
usr.lib.dovecot.deliver       usr.sbin.rsyslogd
usr.lib.dovecot.dict          usr.sbin.smbd
usr.lib.dovecot.dovecot-auth  usr.sbin.tcpdump
usr.lib.dovecot.dovecot-lda   usr.sbin.traceroute
gacanepa@ubuntu:~$
```

If you execute sudo apparmor_status again, you will see a longer list of profiles in complain mode. You can now perform the following operations:

To switch a profile currently in enforce mode to complain mode:

```
sudo aa-complain /path/to/file
```

and the other way around (complain --> enforce):

```
sudo aa-enforce /path/to/file
```

Wildcards are allowed in the above cases. For example,

```
sudo aa-complain /etc/apparmor.d/*
```

will place all profiles inside /etc/apparmor.d into complain mode, whereas

```
sudo aa-enforce /etc/apparmor.d/*
```

will switch all profiles to enforce mode.

To entirely disable a profile, create a symbolic link in the /etc/apparmor.d/disabled directory:

```
sudo ln -s /etc/apparmor.d/profile.name /etc/apparmor.d/disable/
```

For more information on AppArmor, please refer to the official wiki and to the documentation provided by Ubuntu.

Summary

In this article we have gone through the basics of SELinux and AppArmor, two well-known MACs. When to use one or the other? To avoid difficulties, you may want to consider sticking with the one that comes with your chosen distribution. In any event, they will help you place restrictions on processes and access to system resources to increase the security in your servers.

Chapter 17: Access Control Lists (ACLs) and quotas

Access Control Lists (also known as ACLs) are a feature of the Linux kernel that allows to define more fine-grained access rights for files and directories than those specified by regular ugo/rwx permissions. For example, the standard ugo/rwx permissions does not allow to set different permissions for different individual users or groups. With ACLs this is relatively easy to do, as we will see in this article.

Checking file system compatibility with ACLs

To ensure that your file systems are currently supporting ACLs, you should check that they have been mounted using the acl option. To do that, we will use tune2fs for ext2/3/4 file systems as indicated below. Replace /dev/sda1 with the device or file system you want to check:

```
tune2fs -l /dev/sda1 | grep "Default mount options:"
```

(With XFS, Access Control Lists are supported out of the box).

In the following ext4 file system, we can see that ACLs have been enabled for /dev/xvda2:

```
[root@server ~]# tune2fs -l /dev/xvda2 | grep "Default mount options:"
Default mount options:    user_xattr acl
[root@server ~]#
```

If the above command does not indicate that the file system has been mounted with support for ACLs, it is most likely due to the noacl option being present in /etc/fstab. In that case, remove it, unmount the file system, and then mount it again, or simply reboot your system after saving the changes to /etc/fstab.

Introducing ACLs

To illustrate how ACLs work, we will use a group named developers and add users walterwhite and saulgoodman (yes, I am a Breaking Bad fan!) to it.:

```
groupadd developers
useradd walterwhite
useradd saulgoodman
usermod -a -G developers walterwhite
usermod -a -G developers saulgoodman
```

Before we proceed, let's verify that both users have been added to the developers group:

```
id walterwhite
id saulgoodman
```

```
[root@server ~]# id walterwhite
uid=5003(walterwhite) gid=5003(walterwhite) groups=5003(walterwhite),5005(developers)
[root@server ~]# id saulgoodman
uid=5004(saulgoodman) gid=5004(saulgoodman) groups=5004(saulgoodman),5005(developers)
[root@server ~]#
```

Let's now create a directory called test in /mnt, and a file named acl.txt inside (/mnt/test/acl.txt). Then we will set the group owner to developers and change its default ugo/rwx permissions recursively to 770 (thus granting read, write, and execute permissions granted to both the owner and the group owner of the file):

```
mkdir /mnt/test
```

```
touch /mnt/test/acl.txt
chgrp -R developers /mnt/test
chmod -R 770 /mnt/test
```

As expected, you can write to /mnt/test/acl.txt as walterwhite or saulgoodman:

```
su - walterwhite
echo "My name is Walter White" > /mnt/test/acl.txt
exit
su - saulgoodman
echo "My name is Saul Goodman" >> /mnt/test/acl.txt
exit
```

```
[root@server ~]# su - walterwhite
[walterwhite@server ~]$ echo "My name is Walter White" > /mnt/test/acl.txt
[walterwhite@server ~]$ exit
logout
[root@server ~]# su - saulgoodman
[saulgoodman@server ~]$ echo "My name is Saul Goodman" >> /mnt/test/acl.txt
[saulgoodman@server ~]$ exit
logout
[root@server ~]# cat /mnt/test/acl.txt
My name is Walter White
My name is Saul Goodman
[root@server ~]#
```

So far so good. However, we will soon see a problem when we need to grant write access to /mnt/test/acl.txt for another user that is not in the developers group. Standard ugo/rwx permissions would require that the new user be added to the developers group, but that would give him / her the same permissions over all the objects owned by the group. That is precisely where ACLs come in handy.

Setting ACLs

There are two types of ACLs: access ACLs are (which are applied to a file or directory), and default (optional) ACLs, which can only be applied to a directory. If files inside a directory where a default ACL has been set do not have a ACL of their own, they inherit the default ACL of their parent directory.

Let's give user gacanepa read and write access to /mnt/test/acl.txt. Before doing that, let's take a look at the current ACL settings in that directory with

```
getfacl /mnt/test/acl.txt
```

Then change the ACLs on the file, use u: followed by the username and **:rw** to indicate read / write permissions:

```
setfacl -m u:gacanepa:rw /mnt/test/acl.txt
```

And run **getfacl** on the file again to compare. The following image shows the "Before" and "After":

```
[root@server ~]# getfacl /mnt/test/acl.txt
getfacl: Removing leading '/' from absolute path names
# file: mnt/test/acl.txt
# owner: root
# group: developers
user::rwx
group::rwx
other::---

[root@server ~]# setfacl -m u:gacanepa:rw /mnt/test/acl.txt
[root@server ~]# getfacl /mnt/test/acl.txt
getfacl: Removing leading '/' from absolute path names
# file: mnt/test/acl.txt
# owner: root
# group: developers
user::rwx
user:gacanepa:rw-
group::rwx
mask::rwx
other::---
```


Next, we will need to give others execute permissions on the /mnt/test directory:

```
chmod +x /mnt/test
```

Keep in mind that in order to access the contents of a directory, a regular user needs execute permissions on that directory.

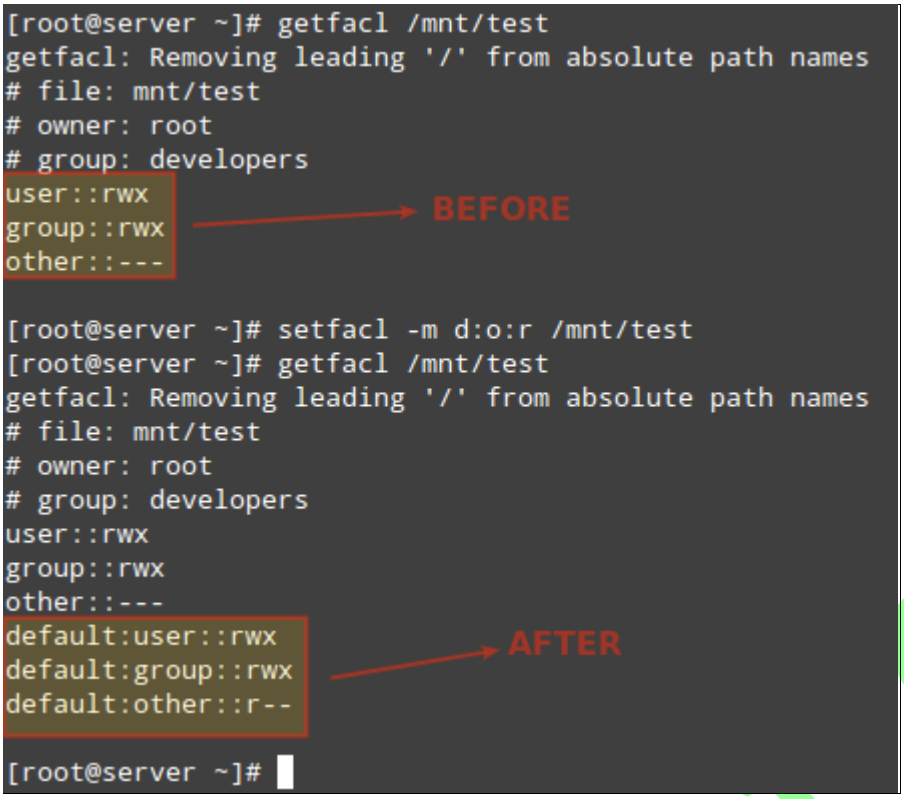
User gacanepa should now be able to write to the file. Switch to that user account and execute the following command to confirm:

```
echo "My name is Gabriel Cánepa" >> /mnt/test/acl.txt
```

To set a default ACL to a directory (which its contents will inherit unless overwritten otherwise), add d: before the rule and specify a directory instead of a file name:

```
setfacl -m d:o:r /mnt/test
```

The ACL above will allow users not in the owner group to have read access to the future contents of the /mnt/test directory. Note the difference in the output of **getfacl /mnt/test** before and after the change:



To remove a specific ACL, replace -m in the commands above with -x. For example,

```
setfacl -x d:o /mnt/test
```

Alternatively, you can also use the -b option to remove ALL ACLs in one step:

```
setfacl -b /mnt/test
```

For more information and examples on the use of ACLs, please refer to chapter 10, section 2, of the openSUSE Security Guide (also available for download at no cost in PDF format).

Disk quotas

Storage space is another resource that must be carefully used and monitored. To do that, quotas can be set on a file system basis, either for individual users or for groups. Thus, a limit is placed on the disk usage allowed for a given user or a specific group, and you can rest assured that your disks will not be filled to capacity by a careless (or malintentioned) user.

The first thing you must do in order to enable quotas on a file system is to mount it with the usrquota or grpquota (for user and group quotas, respectively) options in /etc/fstab. For example, let's enable user-based quotas on /dev/vg00/vol_backups and group-based quotas on /dev/vg00/vol_projects. Note that the UUID is used to identify each file system.

```
UUID=f6d1eba2-9aed-40ea-99ac-75f4be05c05a /home/projects ext4 defaults,grpquota 0 0
UUID=e1929239-5087-44b1-9396-53e09db6eb9e /home/backups ext4 defaults,usrquota 0 0
```

Unmount and remount both file systems:

```
umount /home/projects
umount /home/backups
mount -o remount /home/projects
mount -o remount /home/backups
```

Then check that the **usrquota** and **grpquota** options are present in the output of **mount** (see highlighted below):

```
mount | grep vg00
```

```
[root@server ~]# mount | grep vg00
/dev/mapper/vg00-vol_projects on /home/projects type ext4 (rw,relatime,seclabel,grpquota,data=ordered)
/dev/mapper/vg00-vol_backups on /home/backups type ext4 (rw,relatime,seclabel,quota,usrquota,data=ordered)
[root@server ~]#
```

Finally, run the following commands to initialize and enable quotas:

```
quotacheck -avugc
quotaon -vu /home/backups
quotaon -vg /home/projects
```

That said, let's now assign quotas to the username and group we mentioned earlier. You can later disable quotas with quotaoff.

Setting disk quotas

Let's begin by setting an ACL on /home/backups for user gacanepa, which will give him read, write, and execute permissions on that directory:

```
setfacl -m u:gacanepa:rwx /home/backups/
```

Then with

```
edquota -u gacanepa
```

we will make the soft limit=900 and the hard limit=1000 blocks (1024 bytes/block * 1000 blocks = 1024000 bytes = 1 MB) of disk space usage. We can also place a limit of 20 and 25 as soft and hard limites on the number of files this user can create. The above command will launch the text editor ($EDITOR) with a temporary file where we can set the limits mentioned previously:

```
Disk quotas for user gacanepa (uid 1000):
  Filesystem                   blocks       soft       hard     inodes     soft     hard
  /dev/mapper/vg00-vol_backups          0        900       1000          1       20       25
~
```

These settings will cause a warning to be shown to user gacanepa when he has either reached the 900-block or 20-inode limits for a default grace period of 7 days. If the over-quota situation has not been eliminated by then (for example, by removing files), the soft limit will become the hard limit and this user will be prevented from using more storage space or creating more files.

To test, let's have user gacanepa try to create an empty 2 MB file named test1 inside /home/backups:

```
dd if=/dev/zero of=/home/backups/test1 bs=2M count=1
```

```
[gacanepa@server ~]$ dd if=/dev/zero of=/home/backups/test1 bs=2M count=1
dd: error writing '/home/backups/test1': Disk quota exceeded
1+0 records in
0+0 records out
1024000 bytes (1.0 MB) copied, 0.00728197 s, 141 MB/s
[gacanepa@server ~]$ ls -lh /home/backups/test1
-rw-rw-r--. 1 gacanepa gacanepa 1000K Apr 25 13:19 /home/backups/test1
[gacanepa@server ~]$
```

As you can see, the write operation file fails due to the disk quota having been exceeded. Since only the first 1000 KB are written to disk, the result in this case will most likely be a corrupt file.

Similarly, you can create an ACL for the developers groups in order to give members of that group rwx access to /home/projects:

```
setfacl -m g:developers:rwx /home/projects/
```

And set the quota limits with

```
edquota -g developers
```

Just like we did with user gacanepa earlier.

The grace period can be specified for any number of seconds, minutes, hours, days, weeks, or months by executing

```
edquota -t
```

and updating the values under Block grace period and Inode grace period.

As opposed to block or inode usage (which are set on a user or group-basis), the grace period is set system-wide.

To report quotas, you can use quota -u [user] or quota -g [group] for a quick list or **repquota -v [/path/to/filesystem]** for a more detailed (verbose) and nicely formatted report. Of course, you will want to replace [user], [group], and [/path/to/filesystem] with specific user / group names and file system you want to check.

Summary

In this article we have explained how to set Access Control Lists and disk quotas for users and groups. Using both, you will be able to manage permissions and disk usage more effectively. If you want to learn more about quotas, you can refer to the Quota Mini-HowTo in The Linux Documentation Project.

Chapter 18: Setting up network services

When it comes to setting up and using any kind of network services, it is hard to imagine a scenario that Linux cannot be a part of. In this article we will show how to install the following network services in Linux (each configuration will be covered in upcoming separate articles):

1) NFS (Network File System) server

2) Apache web server

3) Squid proxy server + squidguard

4) Email server (Postfix + Dovecot), and

5) Iptables

In addition, we will want to make sure all of those services are automatically started on boot or on-demand.

We must note that even when you can run all of these network services in the same physical machine or virtual private server, one of the first so-called "rules" of network security tells system administrators to avoid doing so to the extent possible. What is the judgment that supports that statement? It's rather simple: if for some reason a network service is compromised in a machine that runs more than one of them, it can be relatively easy for an attacker to compromise the rest as well.

Now, if you really need to install multiple network services on the same machine (in a test lab, for example), make sure you enable only those that you need at a certain moment, and disable them later.

Before we begin, we need to clarify that the current article (along with the rest in the LFCS and LFCE series) is focused on a performance-based perspective, and thus cannot examine every theoretical detail about the covered topics. We will, however, introduce each topic with the necessary information as a starting point.

In order to use the following network services, you will need to disable the firewall for the time being until we learn how to allow the corresponding traffic through the firewall. Please note that this is NOT recommended for a production setup, but we will do so for learning purposes only.

In a default Ubuntu installation, the firewall should not be active. In openSUSE and CentOS, you will need to explicitly disable it:

```
systemctl stop firewalld
systemctl disable firewalld # or systemctl mask firewalld
```

That being said, let's get started!

Installing a NFS server

NFS in itself is a network protocol, whose latest version is NFSv4. This is the version that we will use throughout this series.

A NFS server is the traditional solution that allows remote Linux clients to mount its shares over a network and interact with those file systems as though they are mounted locally, allowing to centralize storage resources for the network.

CentOS:

```
yum update && yum install nfs-utils
```

Ubuntu:

```
aptitude update && aptitude install nfs-kernel-server
```

openSUSE:

```
zypper refresh && zypper install nfsserver
```

Installing Apache

The Apache web server is a robust and reliable FOSS implementation of a HTTP server. As of the end of October 2014, Apache powers 385 million sites, giving it a 37.45% share of the market. You can use Apache to serve a standalone website or multiple virtual hosts in one machine.

```
yum update && yum install httpd # CentOS
aptitude update && aptitude apache2 # Ubuntu
zypper refresh && zypper apache2 # openSUSE
```

Installing Squid and SquidGuard

Squid is a proxy server and web cache daemon and, as such, acts as intermediary between several client computers and the Internet (or a router connected to the Internet), while speeding up frequent requests by caching web contents and DNS resolution at the same time. It can also be used to deny (or grant) access to certain URLs by network segment or based on forbidden keywords, and keeps a log file of all connections made to the outside world on a per-user basis. Squidguard is a redirector that implements blacklists to enhance squid, and integrates seamlessly with it.

CentOS:

```
yum update
yum install epel-release
yum install squid squidGuard # Note the upper-case G
```

Ubuntu:

```
aptitude update && aptitude install squid3 squidguard
```

openSUSE:

```
zypper refresh && zypper install squid squidGuard # Note the upper-case G
```

Postfix + Dovecot

Postfix is a Mail Transport Agent (MTA). It is the application responsible for routing and delivering email messages from a source to a destination mail servers, whereas dovecot is a widely used IMAP and POP3 email server that fetches messages from the MTA and delivers them to the right user mailbox. Dovecot plugins for several relational database management systems are also available.

```
yum update && yum install postfix dovecot # CentOS
aptitude update && aptitude postfix dovecot-imapd dovecot-pop3d # Ubuntu
zypper refresh && zypper postfix dovecot # openSUSE
```

Iptables

In few words, a firewall is a network resource that is used to manage access to or from a private network, and to redirect incoming and outgoing traffic based on certain rules. Iptables is a tool installed by default in Linux and serves as a frontend to the netfilter kernel module, which is the ultimate responsible for implementing a firewall to perform packet filtering / redirection and network address translation functionalities.

Since iptables is installed in Linux by default, you only have to make sure it is actually running. To do that, we should check that the iptables modules are loaded:

```
lsmod | grep ip_tables
```

If the above command does not return anything, it means the ip_tables module has not been loaded. In that case, run

```
modprobe -a ip_tables
```

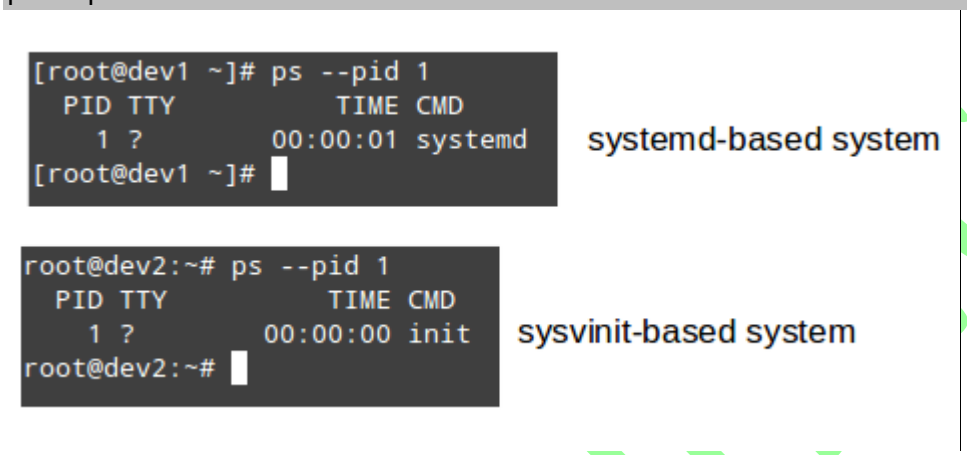
to load the module.

Configuring automatic start on boot

As discussed in Chapter 7 “Managing the startup process and related services”, there are several system and service managers available in Linux. Whatever your choice, you need to know how to start, stop, and restart network services on-demand, and how to enable them to automatically start on boot.

You can check what is your system and service manager by running the following command:

```
ps --pid 1
```



Depending on the output of the above command, you will use one of the following commands to configure whether each service should start automatically on boot or not:

systemd-based:

```
systemctl enable [service] # enable [service] to start at boot
systemctl disable [service] # prevent [service] from starting at boot
sysvinit-based:
chkconfig --level AB [service] on  # Start [service] at boot in runlevels A and B
chkconfig --level CD service off # Don’t start [service] at boot in runlevels C and D
```

upstart-based:

Make sure the /etc/init/[service].conf script exists and contains the minimal configuration, such as:

```
# When to start the service
start on runlevel [2345]
# When to stop the service
stop on runlevel [016]
# Automatically restart process in case of crash
respawn
# Specify the process/command (add arguments if needed) to run
exec /absolute/path/to/network/service/binary arg1 arg2
```

You may also want to check Chapter 7, “Managing the startup process and related services”, (which we just referred to in the beginning of this section) for other useful commands to manage network services on-demand.

Summary

By now you should have all the network services described in this article installed, and possibly running with the default configuration. In later chapters we will explore how to configure them according to our needs.

Chapter 19: The File Transfer Protocol (FTP)

In a day where massive remote storage is rather common, it may be strange to talk about sharing files using FTP (File Transfer Protocol). However, it is still used for file exchange where security does not represent an important consideration and for public downloads of documents, for example. It's for that reason that learning how to configure a FTP server and enable anonymous downloads (not requiring authentication) is still a relevant topic.

In this article we will explain how to set up a FTP server to allow connections on passive mode where the client initiates both channels of communication to the server (one for commands and the other for the actual transmission of files, also known as the control and data channels, respectively). You can read more about passive and active modes (which we will not cover here) in Active FTP vs. Passive FTP, a Definitive Explanation.

That said, let's begin!

Setting up a FTP server

To set up FTP in our server we will install the following packages:

```
yum install vsftpd ftp # CentOS
aptitude install vsftpd ftp # Ubuntu
zypper install vsftpd ftp # openSUSE
```

The vsftpd package is an implementation of a FTP server. The name of the package stands for Very Secure FTP Daemon. On the other hand, ftp is the client program that will be used to access the server. Keep in mind that during the exam, you will be given only one VPS where you will need to install both client and server, so that is precisely the same approach that we will follow in this article.

In CentOS and openSUSE, you will be required to start and enable the vsftpd service:

```
systemctl start vsftpd && systemctl enable vsftpd
```

In Ubuntu, vsftpd should be started and set to start on subsequent boots automatically after the installation. If not, you can start it manually with

```
sudo service vsftpd start
```

Once vsftpd is installed and running, we can proceed to configure our FTP server.

Configuring the FTP server

At any point, you can refer to man vsftpd.conf for further configuration options. We will set the most common options and mention their purpose in this guide.

As with any other configuration file, it is important to make a backup copy of the original before making changes:

```
cp /etc/vsftpd/vsftpd.conf /etc/vsftpd/vsftpd.conf.orig
```

Then open /etc/vsftpd/vsftpd.conf (the main configuration file) and edit the following options as indicated:

1) Make sure you allow anonymous access to the server (we will use the /storage/ftp directory for this example - that's where we will store documents for anonymous users to access) without password:

```
anonymous_enable=YES
no_anon_password=YES
anon_root=/storage/ftp/
```

If you omit the last setting, the ftp directory will default to /var/ftp (the home directory of the dedicated ftp user that was created during installation).

2) To enable read-only access (thus disabling file uploads to the server), set the following variable to NO:

```
write_enable=NO
```

Important: Only use steps #3 and #4 if you choose to disable the anonymous logins.

3) Likewise, you may want to also allow local users to login with their system credentials to the FTP server. Later on this article we will show you how to restrict them to their respective home directories to store and retrieve files using FTP:

```
local_enable=YES
```

If SELinux is in enforcing mode, you will also need to set the ftp_home_dir flag to on so that FTP is allowed to write and read files to and from their home directories:

```
getsebool ftp_home_dir
```

If not, you can enable it permanently with:

```
setsebool -P ftp_home_dir 1
```

The expected output is shown below:

```
[root@ftp ~]# getsebool ftp_home_dir
ftp_home_dir --> off
[root@ftp ~]# setsebool -P ftp_home_dir 1
[root@ftp ~]# getsebool ftp_home_dir
ftp_home_dir --> on
```

4) In order to restrict authenticated system users to their home directories, we will use

```
chroot_local_user=YES
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd/chroot_list
```

With the above chroot settings and an empty /etc/vsftpd/chroot_list file (which YOU need to create), you will restrict ALL system users to their home directories. Please note this still requires that you ensure that none of them has write permissions to the top directory.

If you want to allow a specific user (or more) outside their home directories, insert the usernames in /etc/vsftpd/chroot_list, one per line.

5) In addition, the following settings will allow you to limit the available bandwidth for anonymous logins (10 KB) and authenticated users (20 KB) in bytes per second, and restrict the number of simultaneous connections per IP address to 5:

```
anon_max_rate=10240
local_max_rate=20480
max_per_ip=5
```

6) We will restrict the data channel to TCP ports 15000 through 15500 in the server. Note this is an arbitrary choice and you can use a different range if you wish. Add the following lines to /etc/vsftpd/vsftpd.conf if they are not already present:

```
pasv_enable=YES
pasv_max_port=15000
pasv_min_port=15500
```

7) Finally, you can set a welcome message to be shown each time a user access the server. A little information without further details will do:

```
ftpd_banner=This is a test FTP server brought to you by Tecmint.com
```

8) Now don’t forget to restart the service in order to apply the new configuration:

```
systemctl restart vsftpd # CentOS
sudo service vsftpd restart # Ubuntu
```

9) Allow FTP traffic through the firewall (for firewalld):

```
firewall-cmd --add-service=ftp
firewall-cmd --add-service=ftp --permanent
firewall-cmd --add-port=15000-15500/tcp
firewall-cmd --add-port=15000-15500/tcp --permanent
or iptables:
iptables --append INPUT --protocol tcp --destination-port 21 -m state --state
NEW,ESTABLISHED --jump ACCEPT
iptables --append INPUT --protocol tcp --destination-port 15000:15500  -m state -
-state ESTABLISHED,RELATED --jump ACCEPT
```

Regardless of the distribution, we will need to load the **ip_conntrack_ftp** module:

```
modprobe ip_conntrack_ftp
```

And make it persistent across boots. On CentOS and openSUSE this means adding the module name to the IPTABLES_MODULES in /etc/sysconfig/iptables-config like so:

```
IPTABLES_MODULES="ip_conntrack_ftp"
```

whereas in Ubuntu you'll want to add the module name (without the modprobe command) at the bottom of /etc/modules:

```
echo "ip_conntrack_ftp" >> /etc/modules
```

10) Last but not least, make sure the server is listening on IPv4 or IPv6 sockets (but not both!). We will use IPv4 here:

```
listen=YES
```

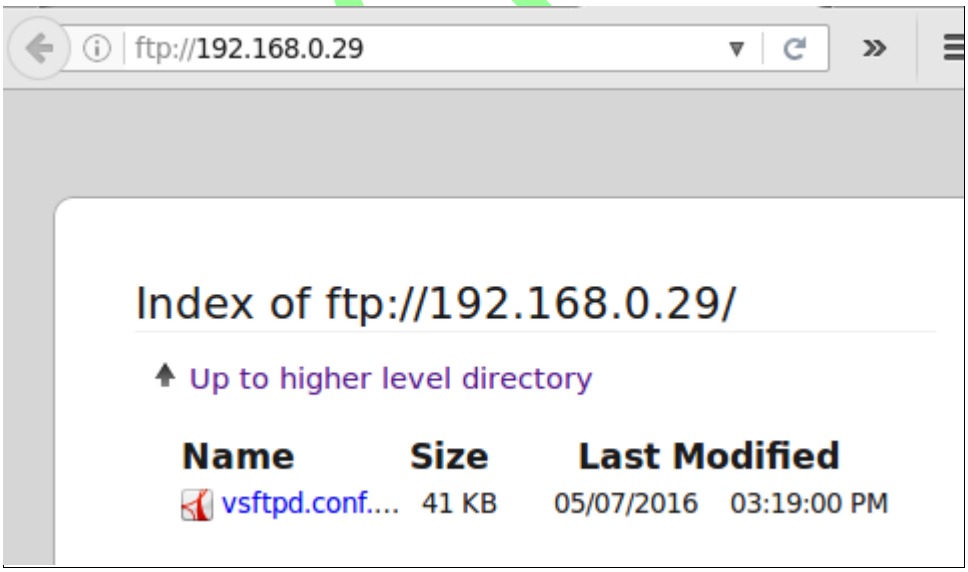
We will now test the newly installed and configured FTP server.

Testing the FTP server

We will create a regular PDF file (in this case, the PDF version of the vsftpd.conf manpage) in /storage/ftp. Note that you may need to install the ghostcript package (which provides ps2pdf) separately, or use another file of your choice:

```
man -t vsftpd.conf | ps2pdf - /storage/ftp/vstpd.conf.pdf
```

To test, we will use both a web browser (by going to ftp://Your_IP_here) and using the command line client (ftp). Let's see what happens when we enter that FTP address in our browser:

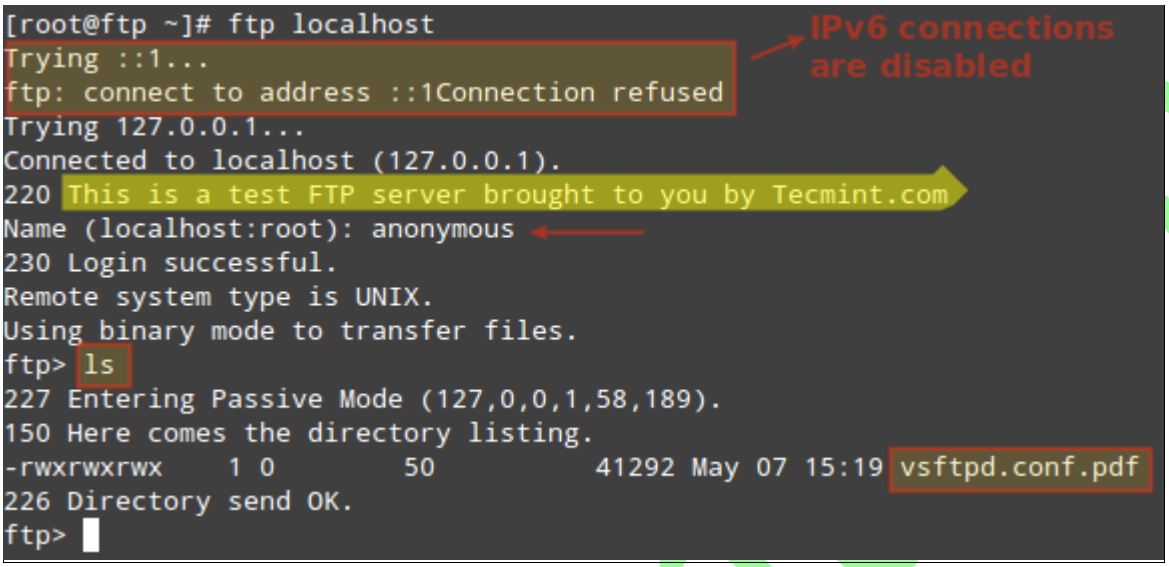


As you can see, the PDF file we saved earlier in /storage/ftp is available for you to download.

On the command line, type

```
ftp localhost
```

And enter anonymous as the user name. You should not be prompted for your password:



To retrieve files using the command line, use the get command followed by the filename, like so:

```
get vsftpd.conf.pdf
```

and you're good to go.

Summary

In this guide we have explained how to properly set up a FTP and use it to allow anonymous logins. You can also follow the instructions given to disable such logins and only allow local users to authenticate using their system credentials (not illustrated in this article since it is not required on the exam).

If you run into any issues, please share with us the output of the following command, which will stripe the configuration file from commented and empty lines, and we will be more than glad to take a look:

```
grep -Eiv '(^$|^#)' /etc/vsftpd/vsftpd.conf
```

Mine is as below (note that there are other configuration directives that we did not cover in this article as they are set by default, so no change was required at our side):

```
local_enable=NO
write_enable=NO
local_umask=022
dirmessage_enable=YES
```

```
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
ftpd_banner=This is a test FTP server brought to you by Tecmint.com
listen=YES
listen_ipv6=NO
pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES
anon_max_rate=10240
local_max_rate=20480
max_per_ip=5
anon_root=/storage/ftp
no_anon_password=YES
allow_writeable_chroot=YES
pasv_enable=YES
pasv_min_port=15000
pasv_max_port=15500
```

Particularly, this directive

```
xferlog_enable=YES
```

will enable the transfer log in /var/log/xferlog. Make sure you look in that file while troubleshooting.

Chapter 20: Setting up a DNS server

Imagine what it would be like if we had to remember the IP addresses of all the websites that we use on a daily basis. Even if we had a prodigious memory, the process to browse to a website would be ridiculously slow and time-consuming. And what about if we needed to visit multiple websites or use several applications that reside in the same machine or virtual host? That would be one of the worst headaches I can think of - not to mention the possibility that the IP address associated with a website or application can be changed without prior notice. Just the very thought of it would be enough reason to desist using the Internet or internal networks after a while.

That's precisely what a world without Domain Name System (also known as DNS) would be. Fortunately, this service solves all of the issues mentioned above - even if the relationship between an IP address and a name changes. For that reason, in this article we will learn how to configure and use a simple DNS server, a service that will allow to translate domain names into IP addresses and vice versa.

Introducing name resolution

For small networks that are not subject to frequent changes, the /etc/hosts file can be used as a rudimentary method of domain name to IP address resolution. With a very simple syntax, this file allows us to associate a name (and / or an alias) with an IP address as follows:

```
[IP address] [name] [alias(es)]
```

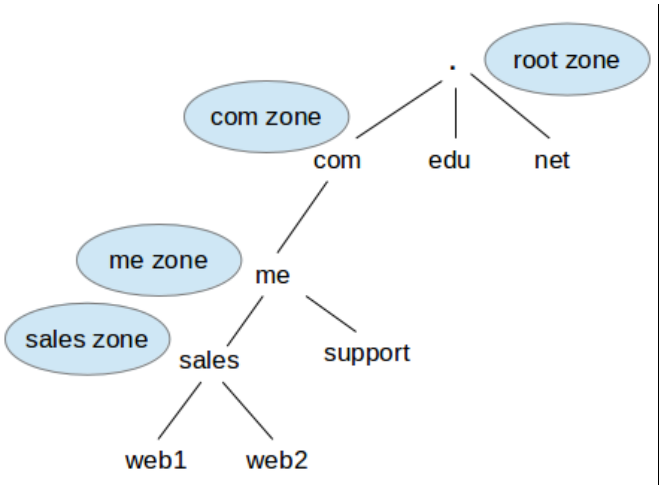
For example,

```
192.168.0.1 gateway gateway.mydomain.com
192.168.0.2 web web.mydomain.com
```

Thus, you can reach the web machine either by its name, the web.mydomain.com alias, or its IP address.

For larger networks, or those that are subject to frequent changes, using the /etc/hosts file to resolve domain names into IP addresses would not be an acceptable solution. That's where the need for a dedicated service comes in.

Under the hood, a DNS server queries a large database in the form of a tree, which starts at the root (".") zone. The following image will help us to illustrate:



In the image above, the root (.) zone contains com, edu, and net domains. Each of these domains are (or can be) managed by different organizations to avoid depending on a big, central one. This allows to properly distribute requests in a hierarchical way.

Let's see what happens under the hood:

1) When a client makes a query to a DNS server for web1.sales.me.com, the server sends the query to the top (root) DNS server, which points the query to the name server in the .com zone. This, in turn, sends the query to the next level name server (in the me.com zone), and then to sales.me.com.This process is repeated as many times as needed until the FQDN (Fully Qualified Domain Name, web1.sales.me.com in this example) is returned by the name server of the zone where it belongs.

2) In this example, the name server in sales.me.com. responds for the address web1.sales.me.com and returns the desired domain name-IP association and other information as well (if configured to do so). All this information is sent to the original DNS server, which then passes it back to the client that requested it in the first place. To avoid repeating the same steps for future identical queries, the results of the query are stored in the DNS server.

These are the reasons why this kind of setup is commonly known as a recursive, caching DNS server.

Installing and configuring a DNS server

In Linux, the most used DNS server is bind (short for Berkeley Internet Name Daemon), which can be installed as follows:

```
yum install bind bind-utils # CentOS
zypper install bind bind-utils # openSUSE
aptitude install bind9 bind9utils # Ubuntu
```

Once we have installed bind and related utilities, let's make a copy of the configuration file before making any changes:

```
cp /etc/named.conf /etc/named.conf.orig # CentOS and openSUSE
cp /etc/bind/named.conf /etc/bind/named.conf.orig # Ubuntu
```

Then let's open named.conf and head over to the options block, where we need to set make sure the following settings are present to configure a recursive, caching server with IP 192.168.0.18/24 that can be accessed only by hosts in the same network (as a security measure). The forwarders settings are used to indicate which name servers should be queried first (in the following example we use Google's name servers) for hosts outside our domain:

```
options {
...
listen-on port 53 { 127.0.0.1; 192.168.0.18};
allow-query         { localhost; 192.168.0.0/24; };
recursion yes;
forwarders {
            8.8.8.8;
            8.8.4.4;
};
…
}
```

Outside the options block we will define our sales.me.com zone (in Ubuntu this is usually done in a separate file called named.conf.local) that maps a domain with a given IP address and a reverse zone to map the IP address to the corresponding domain.

However, the actual configuration of each zone will go in separate files as indicated by the file directive ("master" indicates we will only use one DNS server).

Add the following blocks to named.conf:

```
zone "sales.me.com." IN {
    type master;
    file "/var/named/sales.me.com.zone";
};
zone "0.168.192.in-addr.arpa" IN {
    type master;
    file "/var/named/0.162.198.in-addr.arpa.zone";
};
```

Note that in-addr.arpa (for IPv4 addresses) and ip6.arpa (for IPv6) are conventions for reverse zone configurations.

After saving the above changes to named.conf, we can check for errors as follows:

```
named-checkconf /etc/named.conf
```

If any errors are found, the above command will output an informative message with the cause and the line where they are located. Otherwise, it will not return anything.

Configuring zones

In the files /var/named/sales.me.com.zone and /var/named/0.168.192.in-addr.arpa.zone we will configure the forward (domain → IP address) and reverse (IP address → domain) zones.

Let's tackle the forward configuration first:

0) At the top of the file you will find a line beginning with TTL (short for Time To Live), which specifies how long the cached response should "live" before being replaced by the results of a new query. In the line immediately below, we will reference our domain and set the email address where notifications should be sent (note that the root.sales.me.com means root@sales.me.com).

1) A SOA (Start Of Authority) record indicates that this system is the authoritative nameserver for machines inside the sales.me.com domain. The following settings are required when there are two nameservers (one master and one slave) per domain (although such is not our case since it is not required in the exam, they are presented here for your reference):

- The Serial is used to distinguish one version of the zone definition file from a previous one (where settings could have changed). If the cached response points to a definition with a different serial, the query is performed again instead of feeding it back to the client.
- In a setup with a slave (secondary) nameserver, Refresh indicates the amount of time until the secondary should check for a new serial from the master server. In addition, Retry tells the server how often the secondary should attempt to contact the primary if no response from the primary has been received, whereas Expire indicates when the zone definition in the secondary is no longer valid after the master server could not be reached, and Negative TTL is the time that a Non-existent domain (NXdomain) should be cached.

2) A NS record indicates what is the authoritative DNS server for our domain (referenced by the @ sign at the beginning of the line).

3) An A record (for IPv4 addresses) or an AAAA (for IPv6 addresses) translates names into IP addresses. In the example below:

- dns: 192.168.0.18 (the DNS server itself)
- web1: 192.168.0.29 (a web server inside the sales.me.com zone)
- mail1: 192.168.0.28 (a mail server inside the sales.me.com zone)
- mail2: 192.168.0.30 (another mail server)

4) A MX record indicates the names of the authorized mail transfer agents (MTAs) for this domain. The hostname should be prefaced by a number indicating the priority that the current mail server should have when there are two or more MTAs for the domain (the lower the value, the higher the priority - in the following example, mail1 is the primary whereas mail2 is the secondary MTA).

5) A CNAME record sets an alias (www.web1) for a host (web1).

IMPORTANT: The dot (.) at the end of the names is required.

```
$TTL        604800
```

```
@           IN          SOA         sales.me.com. root.sales.me.com. (
                            2016051101 ; Serial
                            10800 ; Refresh
                            3600  ; Retry
                            604800 ; Expire
                            604800) ; Negative TTL
;
@           IN          NS          dns.sales.me.com.
dns         IN          A           192.168.0.18
web1        IN          A           192.168.0.29
mail1       IN          A           192.168.0.28
mail2       IN          A           192.168.0.30
@           IN          MX          10 mail1.sales.me.com.
@           IN          MX          20 mail2.sales.me.com.
www.web1    IN          CNAME   web1
```

Let's now take a look at the reverse zone configuration (/var/named/0.168.192.in-addr.arpa.zone). The SOA record is the same as in the previous file, whereas the last three lines with a PTR (pointer) record indicate the last octet in the IPv4 address of the mail1, web1, and mail2 hosts (192.168.0.28, 192.168.0.29, and 192.168.0.30, respectively).

```
$TTL        604800
@           IN          SOA         sales.me.com. root.sales.me.com. (
                            2016051101 ; Serial
                            10800 ; Refresh
                            3600  ; Retry
                            604800 ; Expire
                            604800) ; Minimum TTL
@           IN          NS          dns.sales.me.com.
28          IN          PTR         mail1.sales.me.com.
29          IN          PTR         web1.sales.me.com.
30          IN          PTR         mail2.sales.me.com.
```

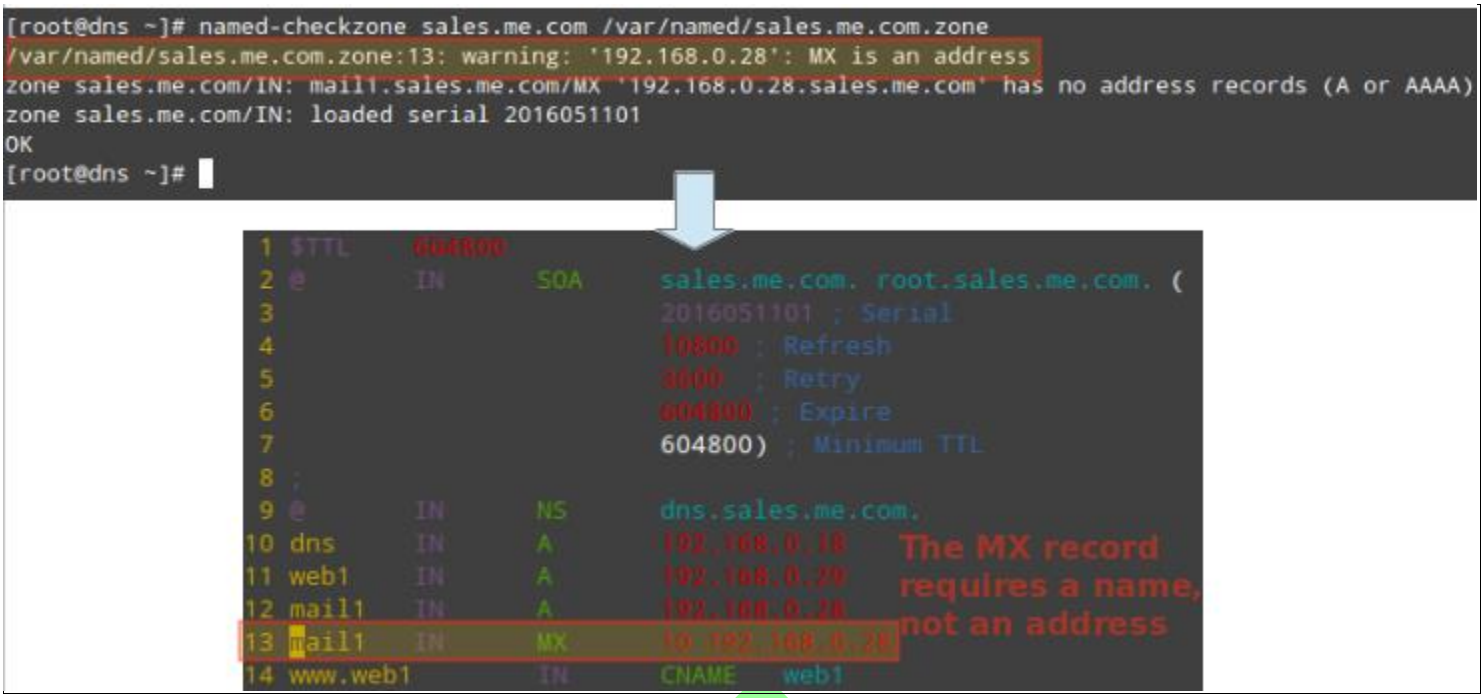
You can check the zone files for errors with:

```
named-checkzone sales.me.com /var/named/sales.me.com.zone
named-checkzone 0.168.192.in-addr.arpa /var/named/0.168.192.in-addr.arpa.zone
```

The following image illustrates what is the expected output on success:

```
[root@dns ~]# named-checkzone sales.me.com /var/named/sales.me.com.zone
zone sales.me.com/IN: loaded serial 2016051101
OK
[root@dns ~]# named-checkzone 0.168.192.in-addr.arpa /var/named/0.168.192.in-addr.arpa.zone
zone 0.168.192.in-addr.arpa/IN: loaded serial 2016051101
OK
[root@dns ~]#
```

Otherwise, you will get an error message stating the cause and how to fix it:



Once you have verified the main configuration file and the zone files, restart the named service to apply changes. In CentOS and openSUSE, do:

```
systemctl restart named
```

And don't forget to enable it as well:

```
systemctl enable named
```

In Ubuntu:

```
sudo service bind9 restart
```

Finally, you will have to edit the configuration of your main network interfaces:

```
DNS1=192.168.0.18 # In /etc/sysconfig/network-scripts/ifcfg-enp0s3 for CentOS and openSUSE
dns-nameservers 192.168.0.18 # in /etc/network/interfaces for Ubuntu
```

and restart the network service to apply changes.

Testing the DNS server

At this point we are ready to query our DNS server for local and outside names and addresses.

The following commands will return the IP address associated with the host web1:

```
host web1.sales.me.com
host web1
host www.web1
```

```
[root@dns ~]# host web1.sales.me.com
web1.sales.me.com has address 192.168.0.29
[root@dns ~]# host web1
web1.sales.me.com has address 192.168.0.29
[root@dns ~]# host www.web1
www.web1.sales.me.com is an alias for web1.sales.me.com.
web1.sales.me.com has address 192.168.0.29
[root@dns ~]#
```

How can we find out who is handling emails for sales.me.com? It's easy to find out - just query the MX records for the domain:

```
host -t mx sales.me.com
```

```
[root@dns ~]# host -t mx sales.me.com
sales.me.com mail is handled by 20 mail2.sales.me.com.
sales.me.com mail is handled by 10 mail1.sales.me.com.
[root@dns ~]#
```

Likewise, let's perform a reverse query. This will help us find out the name behind an IP address:

```
host 192.168.0.28
host 192.168.0.29
```

```
[root@dns ~]# host 192.168.0.28
28.0.168.192.in-addr.arpa domain name pointer mail1.sales.me.com.
[root@dns ~]# host 192.168.0.29
29.0.168.192.in-addr.arpa domain name pointer web1.sales.me.com.
[root@dns ~]#
```

You can try the same operations for outside hosts:

```
host -t mx linux.com
host 8.8.8.8
```

```
[root@dns ~]# host -t mx linux.com
linux.com mail is handled by 15 smtp2.linuxfoundation.org.
linux.com mail is handled by 10 smtp1.linuxfoundation.org.
[root@dns ~]# host 8.8.8.8
8.8.8.8.in-addr.arpa domain name pointer google-public-dns-a.google.com.
[root@dns ~]#
```

To verify that queries are indeed going through our DNS server, let's enable logging

```
rndc querylog
```

And check the /var/log/messages file (in CentOS and openSUSE):

```
[root@dns log]# rndc querylog
[root@dns log]# host -t mx linux.com
linux.com mail is handled by 10 smtp1.linuxfoundation.org.
linux.com mail is handled by 15 smtp2.linuxfoundation.org.
[root@dns log]# host 8.8.8.8
8.8.8.8.in-addr.arpa domain name pointer google-public-dns-a.google.com.
[root@dns log]# tail -f messages
May 11 23:01:01 dns systemd: Created slice user-0.slice.
May 11 23:01:01 dns systemd: Starting user-0.slice.
May 11 23:01:01 dns systemd: Started Session 6 of user root.
May 11 23:01:01 dns systemd: Starting Session 6 of user root.
May 11 23:01:01 dns systemd: Removed slice user-0.slice.
May 11 23:01:01 dns systemd: Stopping user-0.slice.
May 11 23:06:42 dns named[5498]: received control channel command 'querylog'
May 11 23:06:42 dns named[5498]: query logging is now on
May 11 23:06:49 dns named[5498]: client 127.0.0.1#43906 (linux.com): query: linux.com IN MX + (1
May 11 23:06:53 dns named[5498]: client 127.0.0.1#41118 (8.8.8.8.in-addr.arpa): query: 8.8.8.8.i
```

To disable logging, type again

```
rndc querylog
```

In Ubuntu, enabling logging will require adding the following independent block (same level as the options block) to /etc/bind/named.conf:

```
logging {
        channel query_log {
            file "/var/log/bind9/query.log";
            severity dynamic;
            print-category yes;
            print-severity yes;
            print-time yes;
        };
        category queries { query_log; };
};
```

Note that the log file must exist and be writable by named.

Summary

In this article we have explained how to set up a basic recursive, caching DNS server and how to configure zones for a domain. The mystery of name to IP resolution (and vice versa) is not such anymore! To ensure the proper operation of your DNS server, don't forget to allow the service in your firewall (port TCP 53) as follows:

```
firewall-cmd --add-port=53/tcp
firewall-cmd --add-port=53/tcp --permanent
```

Chapter 21: Configuring a database server

A database server is a critical component of the network infrastructure necessary for today's applications. Without the ability to store, retrieve, update, and delete data (when needed), the usefulness and scope of web and desktop apps becomes very limited. In addition, knowing how to install, manage, and configure a database server (so that it operates as expected) is an essential skill that every system administrator must have.

In this article we will briefly review how to install and secure a MariaDB database server and then we will explain how to configure it.

Installing and securing a MariaDB server

In CentOS 7.x, MariaDB replaced MySQL, which still can be found in the Ubuntu (along with MariaDB). The same is true for openSUSE. For brevity, we will only use MariaDB in this tutorial, but please note that -besides having different names and development philosophies-, both Relational DataBase Management Systems (RDBMSs for short) are almost identical. This means that the client-side commands are the same on both MySQL and MariaDB, and the configuration files are named and located in the same places.

To install MariaDB, do:

```
yum update && yum install mariadb mariadb-server # CentOS
sudo aptitude update && sudo aptitude install mariadb-client mariadb-server #
Ubuntu
zypper update && zypper install mariadb mariadb-tools # openSUSE
```

Note that, in Ubuntu, you will be asked to enter a password for the RDBMS root user.

Once the above packages have been installed, make sure the database service is running and has been activated to start on boot (in CentOS and openSUSE you will need to perform this operation manually, whereas in Ubuntu the installation process will have already taken care of it for you):

```
systemctl start mariadb && systemctl enable mariadb # CentOS
systemctl start mysql && systemctl enable mysql # openSUSE
```

Then run the mysql_secure_installation script. This process will allow you to 1) set / reset the password for the RDBMS root user, 2) remove anonymous logins (thus enabling only users with a valid account to log in to the RDBMS), 3) disable root access for machines other than localhost, 4) remove the test database (which anyone

can access), and 5) activate the changes associated with 1 through 4. For a more detailed description of this process, you can refer to the Post installation section in Install MariaDB 5.5 Database in RHEL/CentOS/Fedora and Debian/Ubuntu.

Configuring the database server

The default configuration options are read from the following files in the given order: /etc/mysql/my.cnf, /etc/my.cnf, and ~/.my.cnf. Most often, only /etc/my.cnf exists. It is on this file that we will set the server-wide settings (which can be overridden with the same settings in ~/.my.cnf for each user).

The first thing that we need to note about my.cnf is that settings are organized into categories (or groups) where each category name is enclosed with square brackets. Server system configurations are given in the [mysqld] section, where typically you will find only the first two settings in the table below. The rest are other frequently used options (where indicated, we will change the default value with a custom one of our choosing):

Setting and description	Default value
datadir is the directory where the data files are stored.	datadir=/var/lib/mysql
socket indicates the name and location of the socket file that is used for local client connections. Keep in mind that a socket file is a resource that is utilized to pass information between applications.	socket=/var/lib/mysql/mysql.sock
bind_address is the address where the database server will listen on for TCP/IP connections. If you need your server to listen on more than one IP address, leave out this setting (0.0.0.0 which means it will listen on all IP addresses assigned to this specific host). We will change this to instruct the service to listen only on its main address (192.168.0.13): `bind-address=192.168.0.13`	bind_address=0.0.0.0
port represents the port where the database server will be listening. We will replace the default value(3306) with 20500 (but we need to make sure nothing else is using that port): `port=20500` While some people will argue that security through obscurity is not good practice, changing the default application ports for higher ones is a rudimentary -yet effective- method to discourage port scans.	port=3306
innodb_buffer_pool_size is the buffer pool (in bytes) of memory that is allocated for data and indexes that are accessed frequently when using Innodb (which is the default in MariaDB) or XtraDB as storage engine.	innodb_buffer_pool_size=134217728

We will replace the default value with 256 MB: `innodb_buffer_pool_size=256M`	
skip_name_resolve indicates whether hostnames will be resolved or not on incoming connections. If set to 1, as we will do in this guide, only IP addresses. Unless you require hostnames to determine permissions, it is advisable to disable this variable (in order to speed up connections and queries) by setting its value to 1: `skip_name_resolve=1`	skip_name_resolve=0
query_cache_size represents the size (in bytes) available to the query cache in disk, where the results of SELECT queries are stored for future use when an identical query (to the same database and using the same protocol and same character set) is performed. You should choose a query cache size that matches your needs based on 1) the number of repetitive queries, and 2) the approximate number of records those repetitive queries are expected to return. We will set this value to 100 MB for the time being: `query_cache_size=100M`	query_cache_size=0 (which means it is disabled by default)
max_connections is the maximum number of simultaneous client connections to the server. We will set this value to 30: `max_connections=30` Each connection will use a thread, and thus will consume memory. Take this fact into account while setting max_connections.	max_connections=151
thread_cache_size indicates the numbers of threads that the server allocates for reuse after a client disconnects and frees thread(s) previously in use. In this situation, it is cheaper (performance-wise) to reuse a thread than instantiating a new one. Again, this depends on the number of connections you are expecting. We can safely set this value to half the number of max_connections: `thread_cache_size=15`	thread_cache_size=0 (disabled by default)

In CentOS, we will need to tell SELinux to allow MariaDB to listen on a non-standard port (20500) before restarting the service:

```
yum install policycoreutils-python
semanage port -a -t mysqld_port_t -p tcp 20500
```

Then restart the service.

Checking the configuration

To assist us in checking and tuning the configuration as per our specific needs, we can install mysqltuner (a script that will provide suggestions to improve the performance of our database server and increase its stability):

```
wget https://github.com/major/MySQLTuner-perl/tarball/master
tar xzf master
```

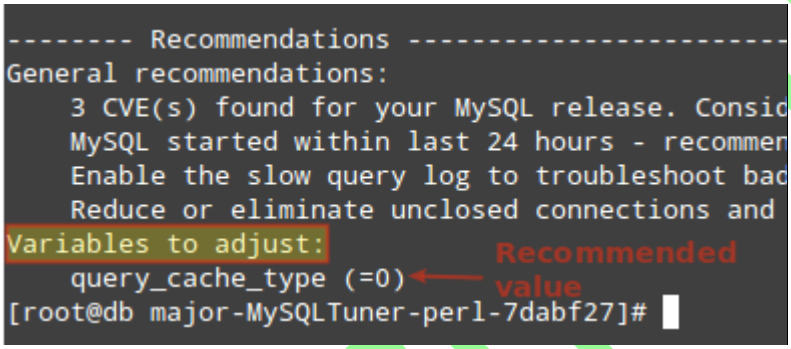
Then change directory into the folder extracted from the tarball (the exact version may differ in your case):

```
cd major-MySQLTuner-perl-7dabf27
```

and run it (you will be prompted to enter the credentials of your administrative MariaDB account)

```
./mysqltuner.pl
```

The output of the script is in itself very interesting, but let's skip to the bottom where the variables to adjust are listed with the recommended value:



The query_cache_type setting indicates whether the query cache is disabled (0) or enabled (1). In this case, mysqltuner is advising us to disable it. So why are we advised to deactivate it now? The reason is that the query cache is useful mostly in high-read / low-write scenarios (which is not our case, since we just installed the database server).

WARNING: Before making changes to the configuration of a production server, you are highly encouraged to consult an expert database administrator to ensure that a recommendation given by mysqltuner will not impact negatively on an existing setting.

Summary

In this article we have explained how to configure a MariaDB database server after we have installed and secured it. The configuration variables listed in the table above are only a few settings that you may want to consider while preparing the server for use or when tuning it later. Always refer to the official MariaDB documentation before making changes.

Chapter 22: Setting up a NFS server

In this chapter we will explain how to properly configure your NFSv4 server (without authentication security) so that you can set up network shares to use in Linux clients as if those file systems were part of the local system. Note that you can use LDAP or NIS for authentication purposes, but both options are out of the scope of the LFCE certification.

Once the NFS server is up and running, we will focus our attention on 1) specifying and configuring the local directories that we want to share over the network, and 2) mounting those network shares in clients automatically, either through the /etc/fstab file or the automount kernel-based utility (autofs). We will explain later when to choose one method or the other.

Before we begin, we need to make sure that the idmapd daemon is running and configured. This service performs the mapping of NFSv4 names (user@mydomain) < -- > to user and group IDs, and is required to implement an NFSv4 server.

Edit /etc/default/nfs-common to enable idmapd:

```
NEED_IDMAPD=YES
```

And edit /etc/idmapd.conf with your local domain name (the default is the FQDN of the host):

```
Domain = yourdomain.com
```

Then start idmapd:

```
service nfs-common start # sysvinit / upstart based systems
systemctl start nfs-common # systemd based systems
```

Exporting network shares

The /etc/exports file contains the main configuration directives for our NFS server, defines the file systems that will be exported to remote hosts and specifies the available options. In this file, each network share is indicated using a separate line, which has the following structure by default:

```
/filesystem/to/export client1([options]) clientN([options])
```

where /filesystem/to/export is the absolute path to the exported file system, whereas client1 (up to clientN) represents the specific client (hostname or IP address) or network (wildcards are allowed) to which the share is being exported. Finally, options is a list of comma-separated values (options) that are taken into account while exporting the share, respectively. Please note that there are no spaces between each hostname and the parentheses it precedes.

Here is a list of the most-frequent options and their respective description:

- **ro** (short for read-only): Remote clients can mount the exported file systems with read permissions only.
- **rw** (short for read-write): Allows remote hosts to make write changes in the exported file systems.

- **wdelay** (short for write delay): the NFS server delays committing changes to disk if it suspects another related write request is imminent. However, if the NFS server receives multiple small unrelated requests, this option will reduce performance, so the no_wdelay option can be used to turn it off.
- **sync**: the NFS server replies to requests only after changes have been committed to permanent storage (i.e., the hard disk). Its opposite, the async option, may increase performance but at the cost of data loss or corruption after an unclean server restart.
- **root_squash**: prevents remote root users from having superuser privileges in the server and assigns them the user ID for user nobody. If you want to "squash" all users (and not just root), you can use the **all_squash** option.
- **anonuid** / **anongid**: explicitly sets the UID and GID of the anonymous account (nobody).
- **subtree_check**: if only a subdirectory of a file system is exported, this option verifies that a requested file is located in that exported subdirectory. On the other hand, if the entire file system is exported, disabling this option with no_subtree_check will speed up transfers. The default option nowadays is no_subtree_check as subtree checking tends to cause more problems than it is worth, according to man 5 exports.
- **fsid=0** | **root** (zero or root): specifies that the specified file system is the root of multiple exported directories (only applies in NFSv4).

In this article we will use the directories /NFS-SHARE and /NFS-SHARE/mydir on 192.168.0.10 (NFS server) as our test file systems.

We can always list the available network shares in a NFS server using the following command:

```
showmount -e [IP or hostname]
```

```
[gacanepa@dev1 ~]$ showmount -e 192.168.0.10
Export list for 192.168.0.10:
/NFS-SHARE/mydir 192.168.0.17
/NFS-SHARE       192.168.0.17
[gacanepa@dev1 ~]$
```

In the output above, we can see that the /NFS-SHARE and /NFS-SHARE/mydir shares on 192.168.0.10 have been exported to client with IP address 192.168.0.17.

Our initial configuration (refer to the /etc/exports directory on your NFS server) for the exported directory is as follows:

```
/NFS-
SHARE          192.168.0.17(fsid=0,no_subtree_check,rw,root_squash,sync,anonuid=1
000,anongid=1000)
/NFS-SHARE/mydir            192.168.0.17(ro,sync,no_subtree_check)
```

After editing the configuration file, we must restart the NFS service:

```
service nfs-kernel-server restart # sysvinit / upstart based systems
systemctl restart nfs-server # systemd based systems
```

Mounting exported network shares using autofs

The downside of mounting a network file system using the mount command or /etc/fstab is that the system must allocate the necessary resources to keep the share mounted at all times, or at least until we decide to unmount them manually. An alternative is to mount the desired file system on-demand automatically (without using the mount command) through autofs, which can mount file systems when they are used and unmount them after a period of inactivity.

Autofs reads /etc/auto.master, which has the following format:

```
[mount point]        [map file]
```

where [map file] is used to indicate multiple mount points within [mount point].

This master map file (/etc/auto.master) is then used to determine which mount points are defined, and then starts an automount process with the specified parameters for each mount point.

Edit your /etc/auto.master as follows:

```
/media/nfs        /etc/auto.nfs-share        --timeout=60
```

and create a map file named /etc/auto.nfs-share with the following contents:

```
writeable_share  -fstype=nfs4 192.168.0.10:/
non_writeable_share  -fstype=nfs4 192.168.0.10:/mydir
```

Note that the first field in /etc/auto.nfs-share is the name of a subdirectory inside /media/nfs. Each subdirectory is created dynamically by autofs.

Now, restart the autofs service:

```
service autofs restart # sysvinit / upstart based systems
systemctl restart autofs # systemd based systems
```

and finally, to enable autofs to start on boot, run the following command:

```
chkconfig --level 345 autofs on
systemctl enable autofs # systemd-based systems
```

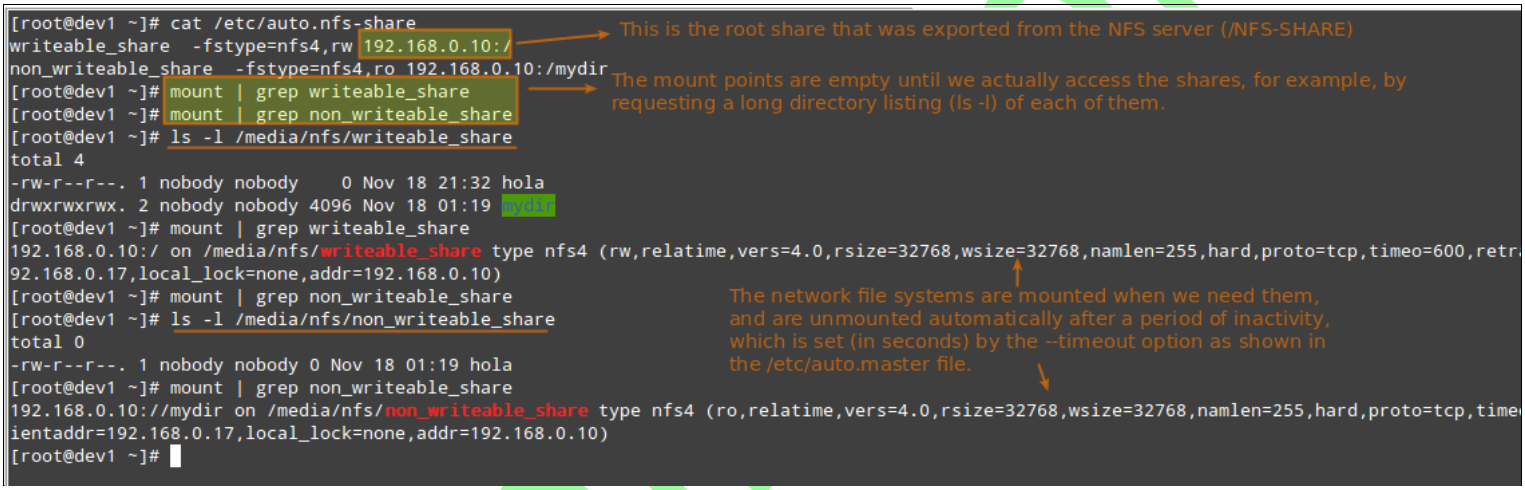
Examining mounted file systems after starting the autofs daemon

When we restart autofs, the mount command shows us that the map file (/etc/auto.nfs-share) is mounted on the specified directory in /etc/auto.master:

```
[root@dev1 ~]# grep nfs-share /etc/auto.master
/media/nfs    /etc/auto.nfs-share    --timeout=60
[root@dev1 ~]#
```

```
[gacanepa@dev1 ~]$ mount | grep nfs-share
/etc/auto.nfs-share on /media/nfs type autofs (rw,relatime,fd=6,pgrp=1172,timeout=60,r
[gacanepa@dev1 ~]$
```

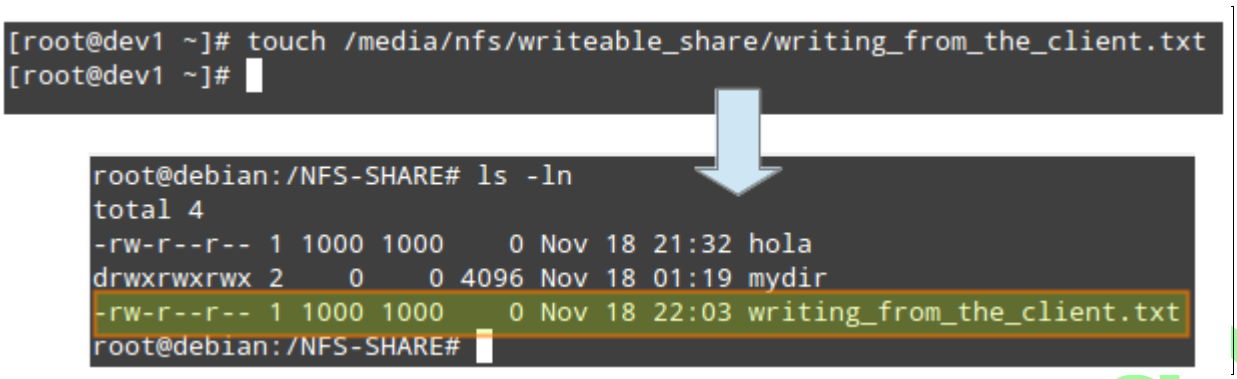
Please note that no directories have actually been mounted yet, but will be automatically when we try to access the shares specified in /etc/auto.nfs-share:



As we can see, the autofs service "mounts" the map file, so to speak, but waits until a request is made to the file systems to actually mount them.

Performing write tests in exported file systems

The anonuid and anongid options, along with the root_squash as set in the first share, allow us to map requests performed by the root user in the client to a local account in the server. In other words, when root in the client creates a file in that exported directory, its ownership will be automatically mapped to the user account with UID and GID = 1000, provided that such account exists on the server:



Summary

I hope you were able to successfully setup and configure a NFS server fit for your environment using this article as a guide. You may also want to refer to the relevant man pages for further help (man exports and man idmapd.conf, for example). Feel free to experiment with other options and test cases as outlined earlier and do not hesitate to use the form below to send your comments, suggestions, or questions. We will be glad to hear from you.

Chapter 23: The Apache web server

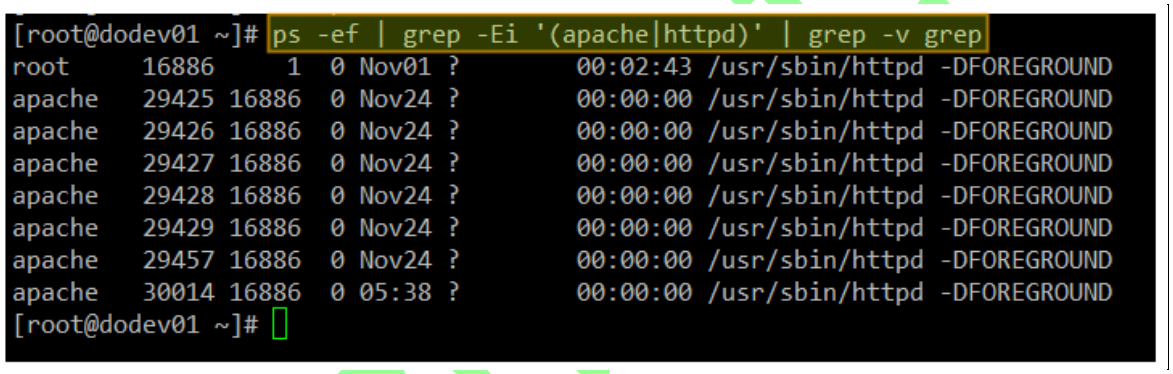
In this chapter we will show you how to configure Apache to serve web content, and how to set up name-based virtual hosts and SSL, including a self-signed certificate. Note that this article is not supposed to be a comprehensive guide on Apache, but rather a starting point for self-study about this topic for the LFCE exam. For that reason we are not covering load balancing with Apache in this tutorial either.

You may already know other ways to perform the same tasks, which is OK considering that the Linux Foundation Certification are strictly performance-based. Thus, as long as you 'get the job done', you stand good chances of passing the exam.

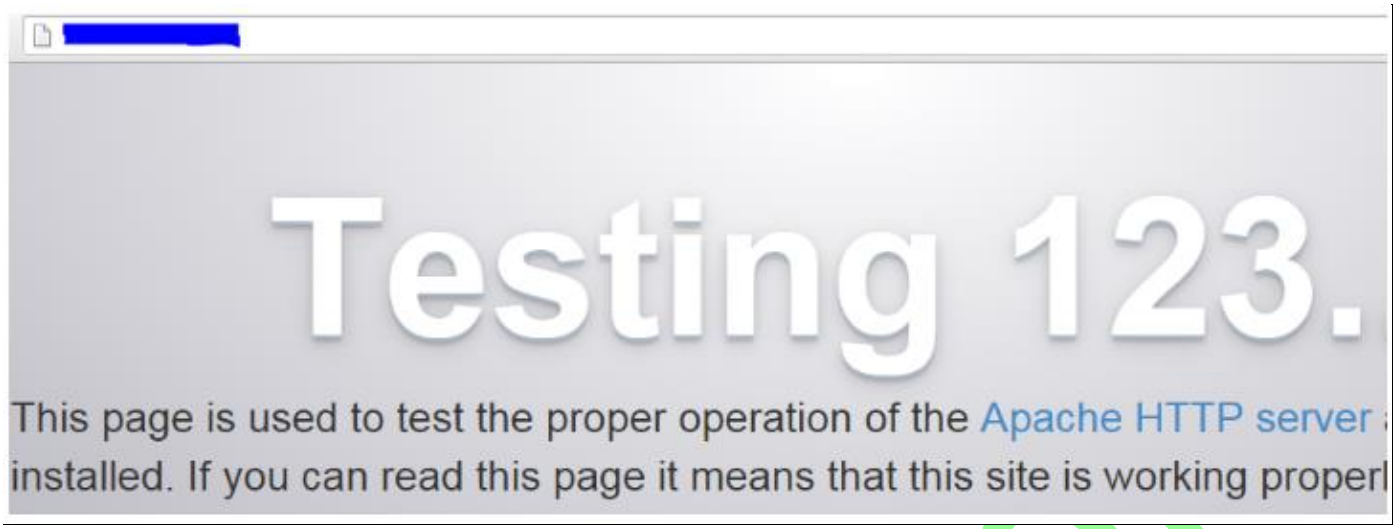
By now, you should have the Apache web server installed and running. You can verify this with the following command:

```
ps -ef | grep -Ei '(apache|httpd)' | grep -v grep
```

Note that the above command checks for the presence of either apache or httpd (the most common names for the web daemon) among the list of running processes. If Apache is running, you will get output similar to the following:



The ultimate method of testing the Apache installation and checking whether it's running is launching a web browser and point to the IP of the server. We should be presented with the following screen or at least a message confirming that Apache is working:



Otherwise, please refer to Chapter 18 for instructions on installing and starting Apache.

Configuring Apache

The main configuration file for Apache can be located in different directories depending on your distribution:

```
/etc/apache2/apache2.conf # Ubuntu
/etc/httpd/conf/httpd.conf # CentOS
/etc/apache2/httpd.conf # openSUSE
```

Fortunately for us, the configuration directives are extremely well documented in the Apache project web site. We will refer to some of them throughout this article.

Serving pages in a standalone web server

The most basic usage of Apache is to serve web pages in a standalone server where no virtual hosts have been configured yet. The **DocumentRoot** directive specifies the directory out of which Apache will serve web pages documents. Note that by default, all requests are taken from this directory, but you can also use symbolic links and / or aliases may be used to point to other locations as well.

Unless matched by the Alias directive (which allows documents to be stored in the local filesystem instead of under the directory specified by DocumentRoot), the server appends the path from the requested URL to the document root to make the path to the document.

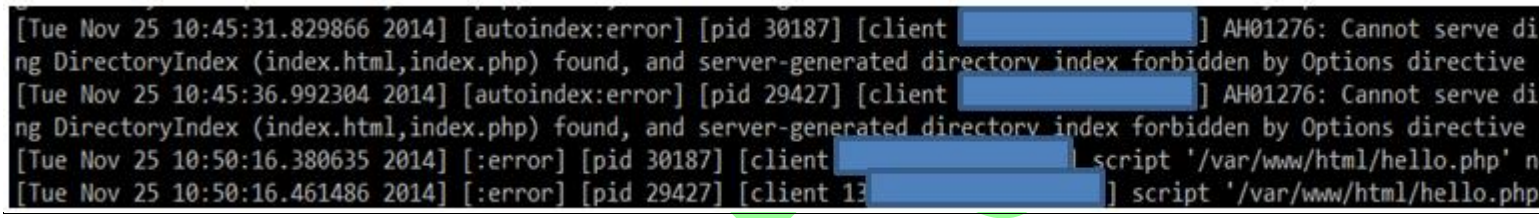
For example, given the following DocumentRoot:

```
DocumentRoot "/var/www/html"
```

When the web browser points to [Server IP or hostname]/lfce/tecmint.html, the server will open /var/www/html/lfce/tecmint.html (assuming that such file exists) and save the event to its access log with a 200 (OK) response. The access log is typically found inside /var/log under a representative name, such as access.log

or access_log. You may even find this log (and the error log as well) inside a subdirectory (for example, /var/log/httpd in CentOS). Otherwise, the failed event will still be logged to the access log but with a 404 (Not Found) response.

In addition, the failed events will be recorded in the error log:



The format of the access log can be customized according to your needs using the LogFormat directive in the main configuration file, whereas you cannot do the same with the error log.

The default format of the access log is as follows:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" [nickname]
```

where each of the letters preceded by a percent sign indicates the server to log a certain piece of information:

String	Description
%h	Remote hostname or IP address
%l	Remote log name
%u	Remote user if the request is authenticated
%t	Date and time when the request was received
%r	First line of request to the server
%>s	Final status of the request
%b	Size of the response [bytes]

and nickname is an optional alias that can be used to customize other logs without having to enter the whole configuration string again.

You may refer to the LogFormat directive [Custom log formats section] in the Apache docs for further options.

Both log files (access and error) represent a great resource to quickly analyze at a glance what's happening on the Apache server. Needless to say, they are the first tool a system administrator uses to troubleshoot issues.

Finally, another important directive is **Listen**, which tells the server to accept incoming requests on the specified port or address/port combination:

- If only a port number is defined, the apache will listens to the given port on all network interfaces (the wildcard sign ***** is used to indicate 'all network interfaces').
- If both IP address and port is specified, then the apache will listen on the combination of given port and network interface.

Please note (as you will see in the examples below) that multiple Listen directives can be used at the same time to specify multiple addresses and ports to listen to. This option instructs the server to respond to requests from any of the listed addresses and ports.

Setting up name-based virtual hosts

The concept of virtual host defines an individual site (or domain) that is served by the same physical machine. Actually, multiple sites / domains can be served off a single "real" server as virtual host. This process is transparent to the end user, to whom it appears that the different sites are being served by distinct web servers.

Name-based virtual hosting allows the server to rely on the client to report the hostname as part of the HTTP headers. Thus, using this technique, many different hosts can share the same IP address.

Each virtual host is configured in a directory within DocumentRoot. For our case, we will use the following dummy domains for the testing setup, each located in the corresponding directory:

- ilovelinux.com - /var/www/html/ilovelinux.com/public_html
- linuxrocks.org - /var/www/html/linuxrocks.org/public_html

In order for pages to be displayed correctly, we will chmod each VH's directory to 755:

```
chmod -R 755 /var/www/html/ilovelinux.com/public_html
chmod -R 755 /var/www/html/linuxrocks.org/public_html
```

Next, create a sample index.html file inside each public_html directory:

```
<html>
  <head>
    <title>www.ilovelinux.com</title>
  </head>
  <body>
    <h1>This is the main page of www.ilovelinux.com</h1>
  </body>
</html>
```

Finally, in CentOS and openSUSE add the following section at the bottom of /etc/httpd/conf/httpd.conf or /etc/apache2/httpd.conf, respectively, or just modify it if it's already there:

```
<VirtualHost *:80>
     ServerAdmin admin@ilovelinux.com
```

```
    DocumentRoot /var/www/html/ilovelinux.com/public_html
    ServerName www.ilovelinux.com
    ServerAlias www.ilovelinux.com ilovelinux.com
    ErrorLog /var/www/html/ilovelinux.com/error.log
    LogFormat "%v %l %u %t \"%r\" %>s %b" myvhost
    CustomLog /var/www/html/ilovelinux.com/access.log        myvhost
</VirtualHost>
<VirtualHost *:80>
    ServerAdmin admin@linuxrocks.org
    DocumentRoot /var/www/html/linuxrocks.org/public_html
    ServerName www.linuxrocks.org
    ServerAlias www.linuxrocks.org linuxrocks.org
    ErrorLog /var/www/html/linuxrocks.org/error.log
    LogFormat "%v %l %u %t \"%r\" %>s %b" myvhost
    CustomLog /var/www/html/linuxrocks.org/access.log        myvhost
</VirtualHost>
```

Please note that you can also add each virtual host definition in separate files inside the /etc/httpd/conf.d directory. If you choose to do so, each configuration file must be named as follows:

```
/etc/httpd/conf.d/ilovelinux.com.conf
/etc/httpd/conf.d/linuxrocks.org.conf
```

In other words, you need to add .conf to the site or domain name.

In Ubuntu, each individual configuration file is named /etc/apache2/sites-available/[site name].conf. Each site is then enabled or disabled with the a2ensite or a2dissite commands, respectively, as follows:

```
a2ensite /etc/apache2/sites-available/ilovelinux.com.conf
a2dissite /etc/apache2/sites-available/ilovelinux.com.conf
a2ensite /etc/apache2/sites-available/linuxrocks.org.conf
a2dissite /etc/apache2/sites-available/linuxrocks.org.conf
```

The a2ensite and a2dissite commands create links to the virtual host configuration file and place (or remove) them in the /etc/apache2/sites-enabled directory.

To be able to browse to both sites from another Linux box, you will need to add the following lines in the /etc/hosts file in that machine in order to redirect requests to those domains to a specific IP address:

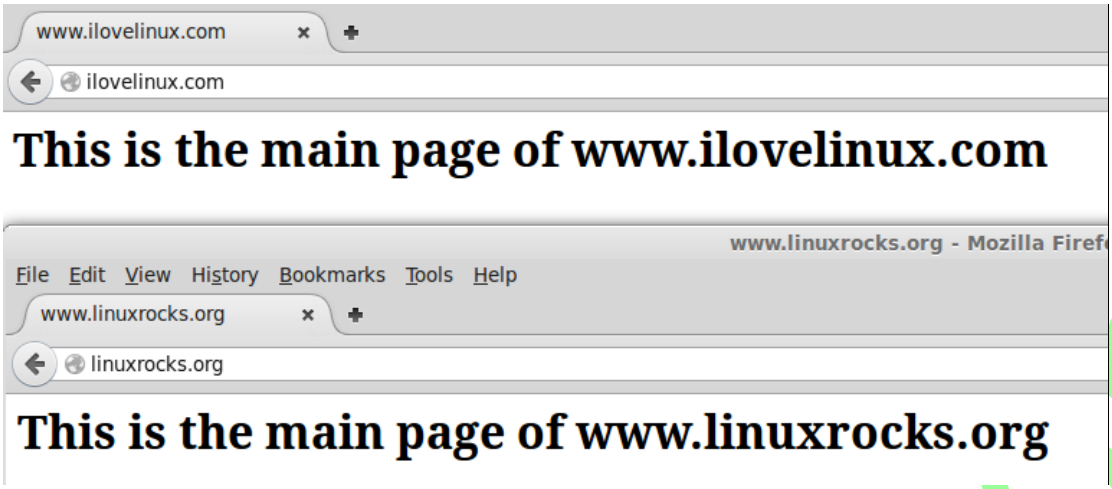
```
[IP address of your web server]        www.ilovelinux.com
[IP address of your web server]        www.linuxrocks.org
```

As a security measure, SELinux will not allow Apache to write logs to a directory other than the default /var/log/httpd. You can either disable SELinux, or set the right security context:

```
chcon system_u:object_r:httpd_log_t:s0 /var/www/html/xxxxxx/error.log
```

where xxxxxx is the directory inside /var/www/html where you have defined your Virtual Hosts.

After restarting Apache, you should see the following page at the above addresses:



INSTALLING AND CONFIGURING SSL WITH APACHE

Finally, we will create and install a self-signed certificate to use with Apache. This kind of setup is acceptable in small environments, such as a private LAN. However, if your server will expose content to the outside world over the Internet, you will want to install a certificate signed by a 3rd party to corroborate its authenticity. Either way, a certificate will allow you to encrypt the information that is transmitted to, from, or within your site.

In CentOS and openSUSE, you need to install the mod_ssl package:

yum update && yum install mod_ssl # CentOS

zypper refresh && zypper install mod_ssl # openSUSE

whereas in Ubuntu you'll have to enable the ssl module for Apache:

a2enmod ssl

The following steps are explained using a CentOS test server, but your setup should be almost identical in the other distributions (if you run into any kind of issues, don't hesitate to leave your questions using the comments form).

Step 1 [Optional]: Create a directory to store your certificates

mkdir /etc/httpd/ssl-certs

Step 2: Generate your self signed certificate and the key that will protect it

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/httpd/ssl-certs/apache.key -out /etc/httpd/ssl-certs/apache.crt

A brief explanation of the options listed above:

- req -X509 indicates we are creating a x509 certificate.

- -nodes (NO DES) means “don’t encrypt the key”.
- -days 365 is the number of days the certificate will be valid for.
- -newkey rsa:2048 creates a 2048-bit RSA key.
- -keyout /etc/httpd/ssl-certs/apache.key is the absolute path of the RSA key.
- -out /etc/httpd/ssl-certs/apache.crt is the absolute path of the certificate.

```
[root@dodev01 ssl-certs]# openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
e.crt
Generating a 2048 bit RSA private key
...........+++
.....................................................................+++
writing new private key to '/etc/httpd/ssl-certs/apache.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:AR
State or Province Name (full name) []:San Luis
Locality Name (eg, city) [Default City]:Villa Mercedes
Organization Name (eg, company) [Default Company Ltd]:GabrielCanepa.com.ar
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:dodev01.gabrielcanepa.com.ar
Email Address []:gacanepa@[redacted].com
[root@dodev01 ssl-certs]#
```

Step 3: Open your chosen virtual host configuration file (or its corresponding section in /etc/httpd/conf/httpd.conf as explained earlier) and add the following lines to a virtual host declaration listening on port 443:

SSLEngine on

SSLCertificateFile /etc/httpd/ssl-certs/apache.crt

SSLCertificateKeyFile /etc/httpd/ssl-certs/apache.key

Please note that you need to add

NameVirtualHost *:443

at the top, right below

NameVirtualHost *:80

Both directives instruct apache to listen on ports 443 and 80 of all network interfaces.

The following example is taken from /etc/httpd/conf/httpd.conf:

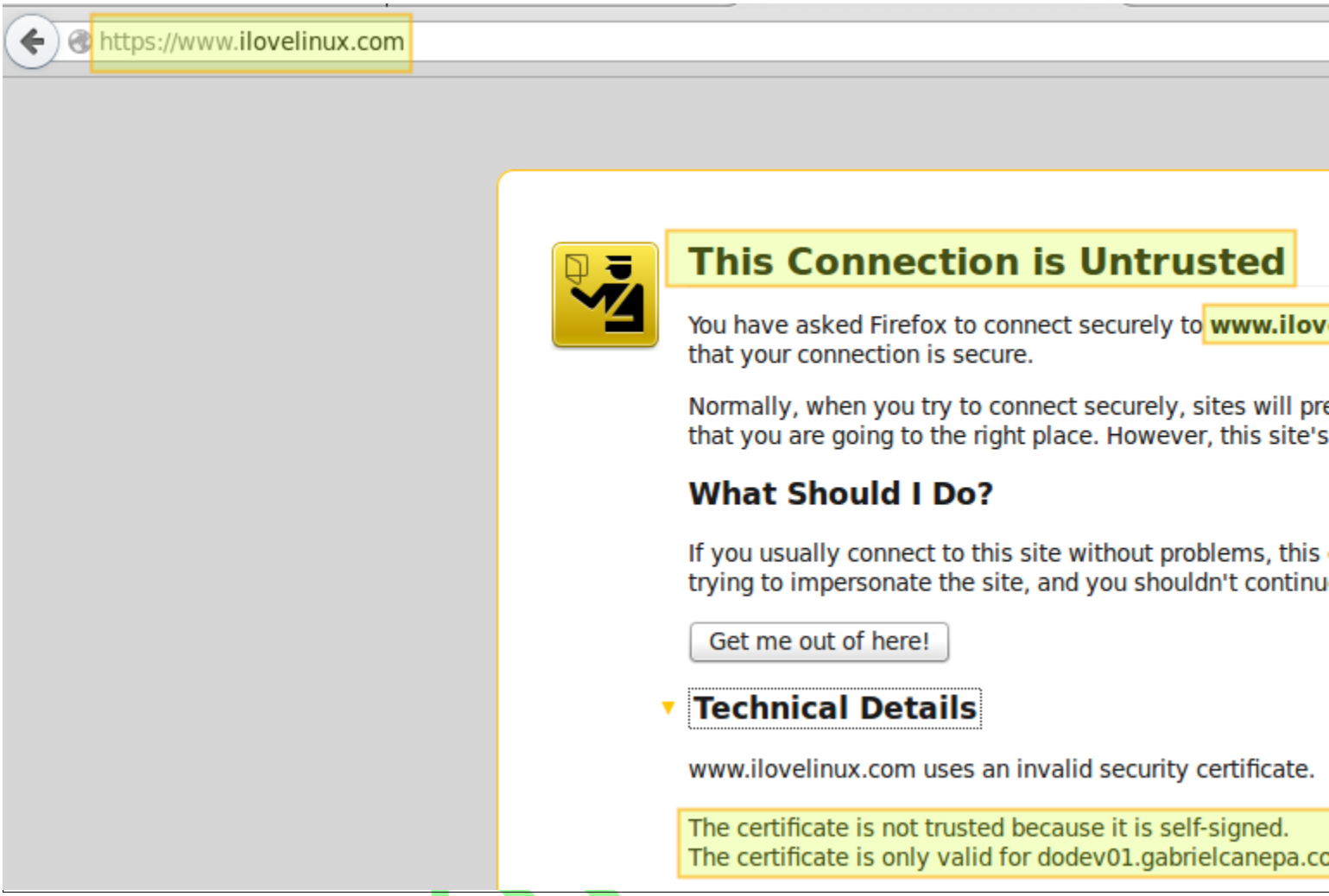
```
NameVirtualHost *:80
NameVirtualHost *:443
<VirtualHost  *:80>
    ServerAdmin admin@ilovelinux.com
    ServerAlias www.ilovelinux.com ilovelinux.com
    DocumentRoot /var/www/html/ilovelinux.com/public_html
    ServerName ilovelinux.com
    ErrorLog logs/ilovelinux.com-error_log
    CustomLog logs/ilovelinux.com-access_log common
</VirtualHost>
<VirtualHost  *:443>
    ServerAdmin admin@ilovelinux.com
    ServerAlias www.ilovelinux.com ilovelinux.com
    DocumentRoot /var/www/html/ilovelinux.com/public_html
    ServerName ilovelinux.com
    ErrorLog /var/www/html/ilovelinux.com/error_log
    LogFormat "%v %l %u %t \"%r\" %>s %b" myvhost
    CustomLog /var/www/html/ilovelinux.com/access_log    myvhost
    SSLEngine on
    SSLCertificateFile /etc/httpd/ssl-certs/apache.crt
    SSLCertificateKeyFile /etc/httpd/ssl-certs/apache.key
</VirtualHost>
```

Then restart Apache

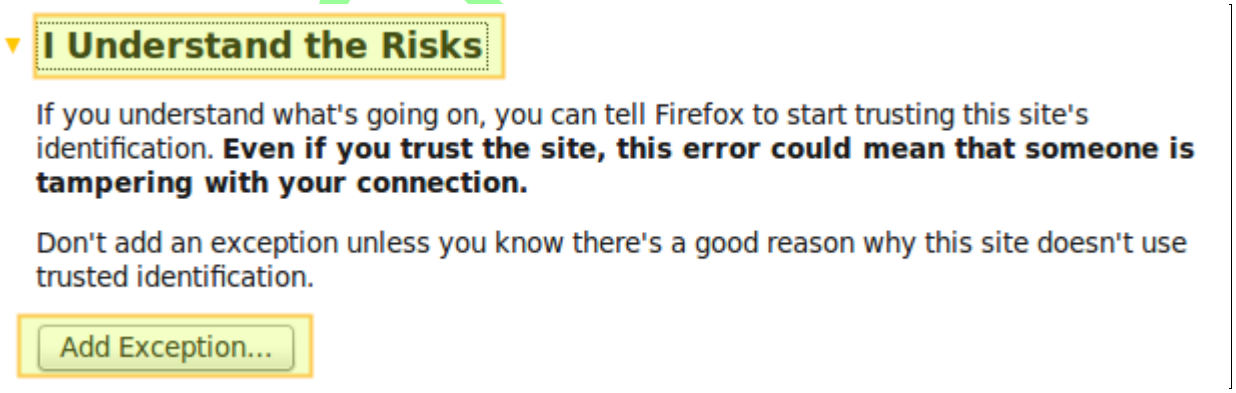
service apache2 restart # sysvinit and upstart based systems

systemctl restart httpd.service # systemd-based systems

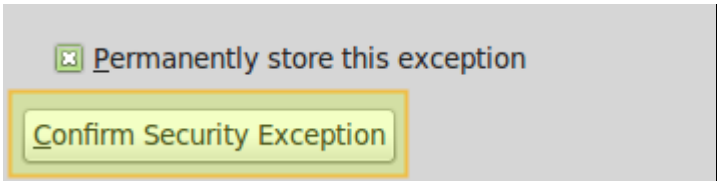
And point your browser to https://www.ilovelinux.com. You will be presented with the following screen:



Go ahead and click on "I understand the risks" and "Add exception":



Finally, check "Permanently store this exception" and click "Confirm Security Exception":



and you will be redirected to your home page using https:

SUMMARY

In this post we have shown how to configure Apache and name-based virtual hosting with SSL to secure data transmission. If for some reason you ran into any issues, feel free to let us know using the comment form below. We will be more than glad to help you perform a successful set up.

Chapter 24: Squid

In this chapter we will show you how to configure the Squid proxy server in order to grant or restrict Internet access, and how to configure an http client, or web browser, to use that proxy server.

Let us remember that, in simple terms, a web proxy server is an intermediary between one (or more) client computers and a certain network resource, the most common being access to the Internet. In other words, the proxy server is connected on one side directly to the Internet (or to a router that is connected to the Internet) and on the other side to a network of client computers that will access the World Wide Web through it.

You may be wondering, why would I want to add yet another piece of software to my network infrastructure?

Here are the top 3 reasons:

1. Squid stores files from previous requests to speed up future transfers. For example, suppose client1 downloads CentOS-7.0-1406-x86_64-DVD.iso from the Internet. When client2 requests access to the same file, squid can transfer the file from its cache instead of downloading it again from the Internet. As you can guess, you can use this feature to speed up data transfers in a network of computers that require frequent updates of some kind.
2. ACLs (Access Control Lists) allow us to restrict the access to websites, and / or monitor the access on a per user basis. You can restrict access based on day of week or time of day, or domain, for example.
3. Bypassing web filters is made possible through the use of a web proxy to which requests are made and which returns requested content to a client, instead of having the client request it directly to the Internet. For example, suppose you are logged on in client1 and want to access www.facebook.com through your company's router. Since the site may be blocked by your company's policies, you can instead connect to a web proxy server and have it request access to www.facebook.com. Remote content is then returned to you through the web proxy server again, bypassing your company's router's blocking policies.

Configuring Squid – the basics

The access control scheme of the Squid web proxy server consists of two different components:

- The ACL elements are directive lines that begin with the word "acl" and represent types of tests that are performed against any request transaction.
- The access list rules consist of an allow or deny action followed by a number of ACL elements, and are used to indicate what action or limitation has to be enforced for a given request. They are checked in order, and list searching terminates as soon as one of the rules is a match. If a rule has multiple ACL elements, it is implemented as a boolean AND operation (all ACL elements of the rule must be a match in order for the rule to be a match).

Squid's main configuration file is /etc/squid/squid.conf, which is ~5000 lines long since it includes both configuration directives and documentation. For that reason, we will create a new squid.conf file with only the lines that include configuration directives for our convenience, leaving out empty or commented lines. To do so, we will use the following commands:

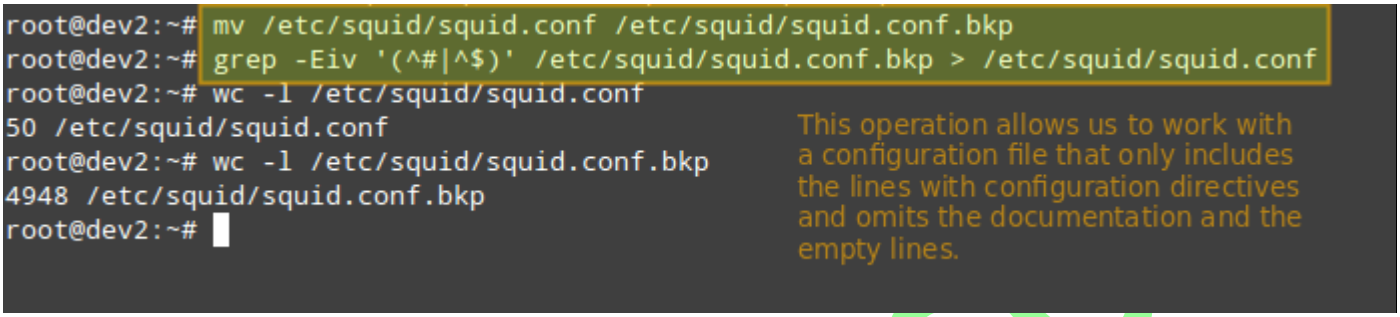
```
mv /etc/squid/squid.conf /etc/squid/squid.conf.bkp
```

And then

```
grep -Eiv '(^#|^$)' /etc/squid/squid.conf.bkp
```

or

```
grep -ve ^# -ve ^$ /etc/squid/squid.conf.bkp > /etc/squid/squid.conf
```



Now, open the newly created squid.conf file, and look for (or add) the following ACL elements and access lists:

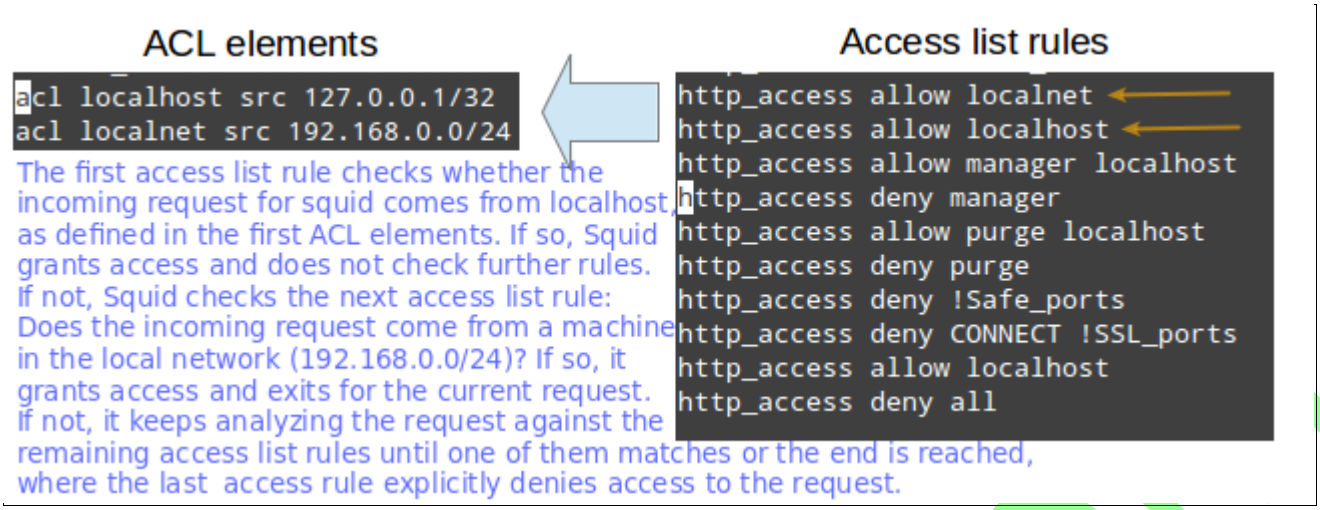
```
acl localhost src 127.0.0.1/32
acl localnet src 192.168.0.0/24
```

The two lines above represent a basic example of the usage of ACL elements:

1. The first word, acl, indicates that this is a ACL element directive line.
2. The second word, localhost or localnet, specify a name for the directive.
3. The third word, src in this case, is an ACL element type that is used to represent a client IP address or range of addresses, respectively. You can specify a single host by IP (or hostname, if you have some sort of DNS resolution implemented) or by network address.
4. The fourth parameter is a filtering argument that is “fed” to the directive.

```
http_access allow localnet
http_access allow localhost
```

The two lines above are access list rules and represent an explicit implementation of the ACL directives mentioned earlier. In few words, they indicate that http access should be granted if the request comes from the local network (localnet), or from localhost. Specifically what is the allowed local network or local host addresses? The answer is: those specified in the localhost and localnet directives.



At this point you can restart Squid in order to apply any pending changes

```
service squid restart # Upstart / sysvinit-based distributions
systemctl restart squid.service # systemd-based distributions
```

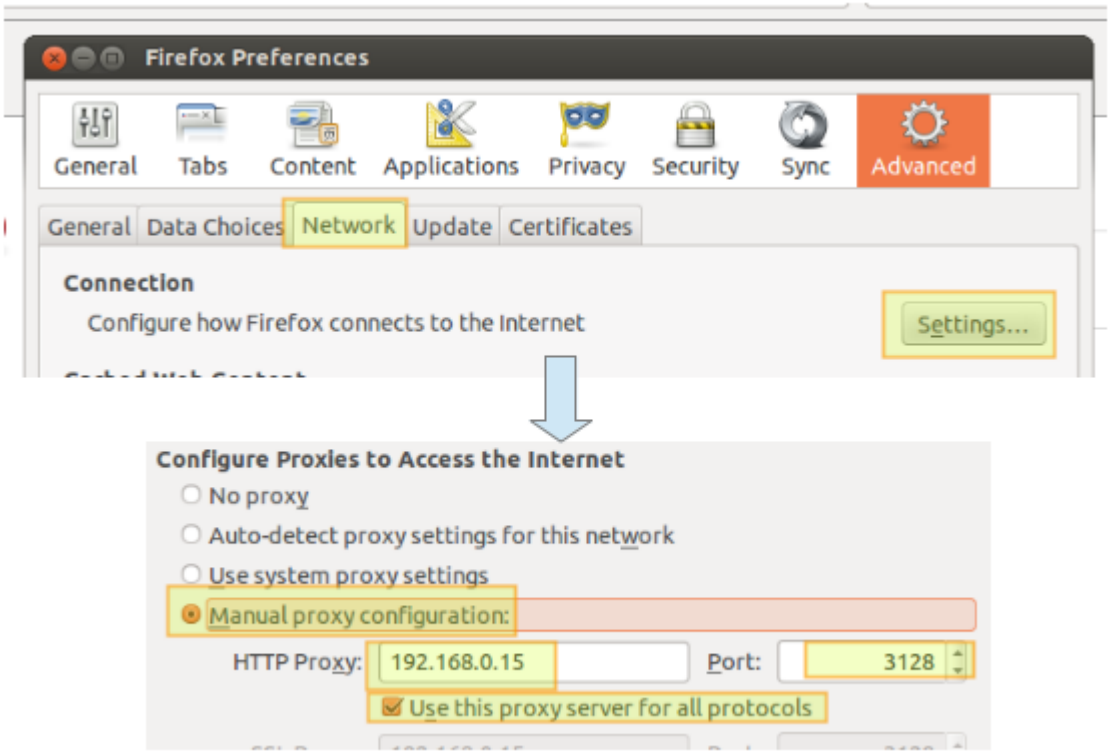
and then configure a client browser in the local network (192.168.0.104 in our case) to access the Internet through your proxy as follows:

In Firefox,

1) Go to the Edit menu and choose the Preferences option.

2) Click on Advanced, then on the Network tab, and finally on Settings...

3) Check Manual proxy configuration and enter the IP address of the proxy server and the port where it is listening for connections.



Note that by default, Squid listens on port 3128, but you can override this behavior by editing the access list rule that begins with http_port (by default it reads http_port 3128).

4) Click OK to apply the changes and you're good to go.

Verifying that a client can access the Internet

You can now verify that your local network client is accessing the Internet through your proxy as follows:

1) In your client, open up a terminal and type

```
ip address show eth0 | grep -Ei '(inet.*eth0)'
```

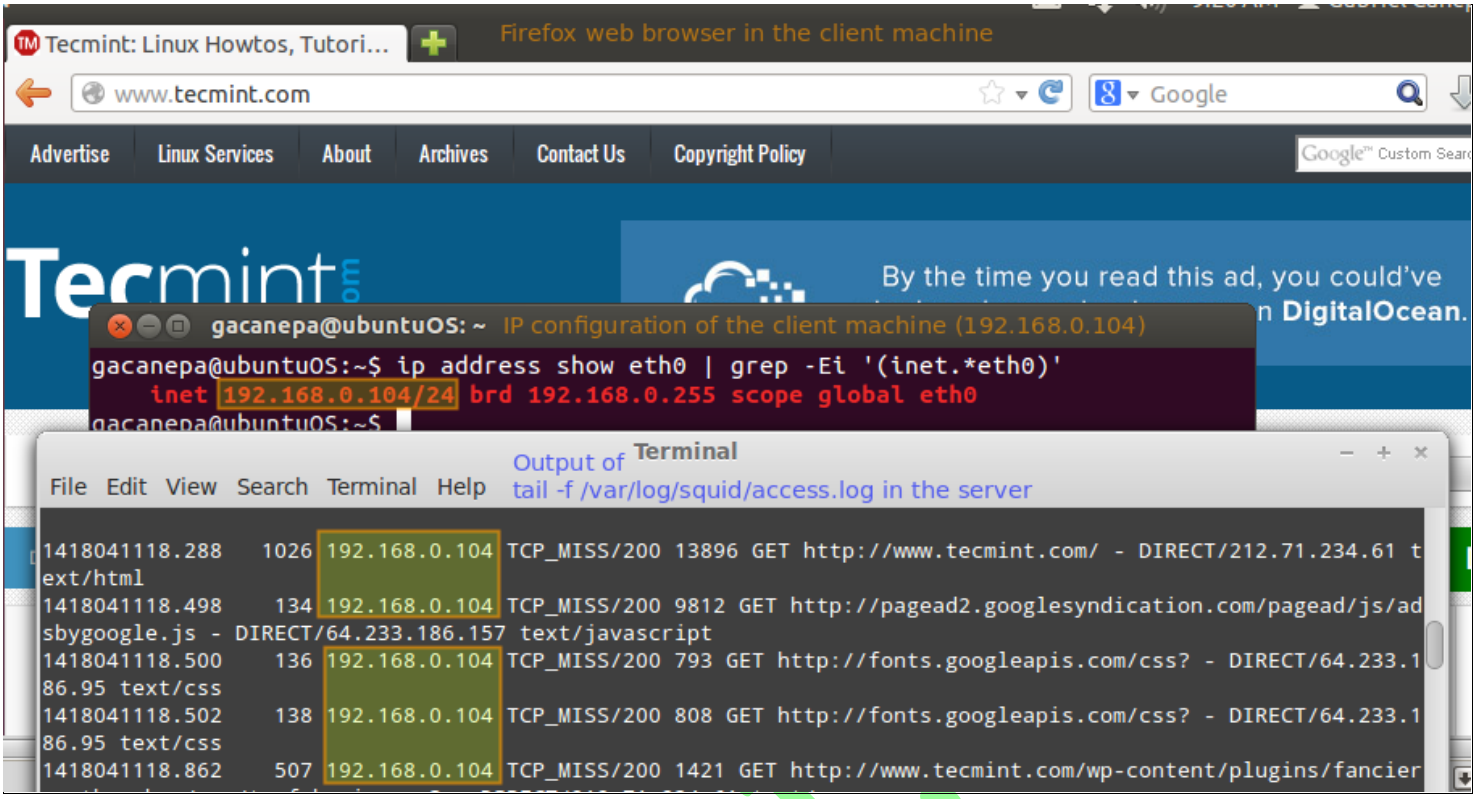
That command will display the current IP address of your client (192.168.0.104 in the following image).

2) In your client, use a web browser to open any given web site (www.tecmint.com in this case).

3) In the server, run

```
tail -f /var/log/squid/access.log
```

and you'll get a live view of requests being served through Squid:



Restricting access by client

Now suppose you want to explicitly deny access to that particular client IP address, while yet maintaining access for the rest of the local network.

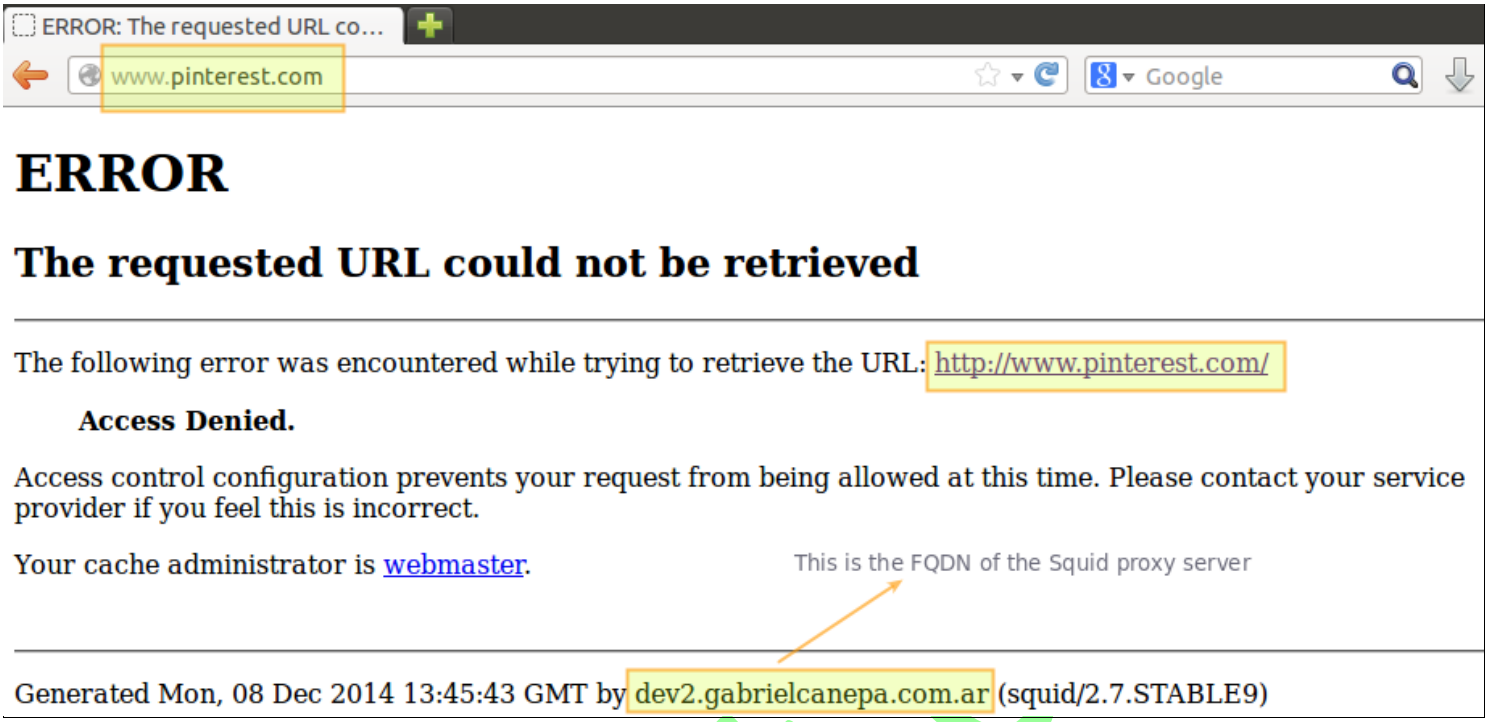
1) Define a new ACL directive as follows (I've named it ubuntuOS but you can name it whatever you want)

```
acl ubuntuOS src 192.168.0.104
```

2) Add the ACL directive to the localnet access list that is already in place, but prefacing it with an exclamation sign. This means, "Allow Internet access to clients matching the localnet ACL directive except to the one that matches the ubuntuOS directive":

```
http_access allow localnet !ubuntuOS
```

3) Now we need to restart Squid in order to apply changes. Then if we try to browse to any site we will find that access is denied now:



Configuring Squid – Fine tuning

To restrict access to Squid by domain we will use the dstdomain keyword in a ACL directive, as follows:

```
acl forbidden dstdomain "/etc/squid/forbidden_domains"
```

where **forbidden_domains** is a plain text file that contains the domains that we desire to deny access to:

```
root@dev2:~# cat /etc/squid/forbidden_domains
.facebook.com
.twitter.com
.pinterest.com
root@dev2:~#
```

One domain per line. The leading . is used to indicate subdomains as well.

Finally, we must grant access to Squid for requests not matching the directive above:

```
http_access allow localnet !forbidden
```

Or maybe we will only want to allow access to those sites during a certain time of the day (10:00 until 11:00 am) only on Monday (M), Wednesday (W), and Friday (F).

```
acl someDays time MWF 10:00-11:00
http_access allow forbidden someDays
http_access deny forbidden
```

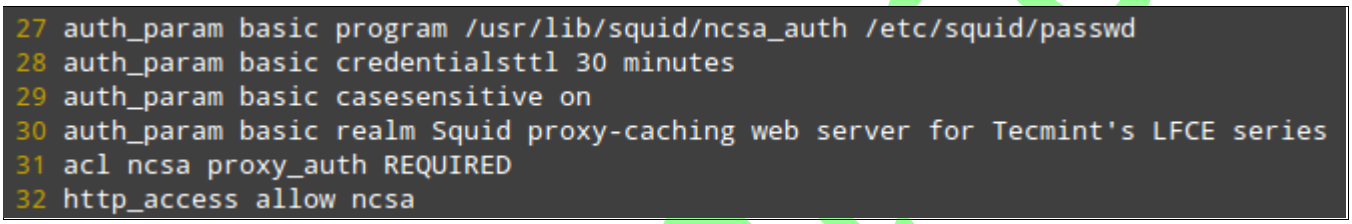
Otherwise, access to those domains will be blocked.

Restricting access by user authentication

Squid support several authentication mechanisms (Basic, NTLM, Digest, SPNEGO, and Oauth) and helpers (SQL database, LDAP, NIS, NCSA, to name a few). In this tutorial we will use Basic authentication with NCSA.

Add the following lines to your /etc/squid/squid.conf file (in CentOS 7, the NCSA plugin will be found in /usr/lib64/squid/basic_nsca_auth).

```
auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/passwd
auth_param basic credentialsttl 30 minutes
auth_param basic casesensitive on
auth_param basic realm Squid proxy-caching web server for Tecmint's LFCE series
acl ncsa proxy_auth REQUIRED
http_access allow ncsa
```



A few clarifications:

- We need to tell Squid which authentication helper program to use with the auth_param directive by specifying the name of the program (most likely, /usr/lib/squid/ncsa_auth), plus any command line options (/etc/squid/passwd in this case) if necessary.
- The /etc/squid/passwd file is created through htpasswd, a tool to manage basic authentication through files. It will allow us to add a list of usernames (and their corresponding passwords) that will be allowed to use Squid.
- credentialsttl 30 minutes will require entering your username and password every 30 minutes (you can specify this time interval with hours as well).
- casesensitive on indicates that usernames and passwords are case sensitive.
- realm represents the text of the authentication dialog that will be used to authenticate to squid.
- Finally, access is granted only when proxy authentication (proxy_auth REQUIRED) succeeds.

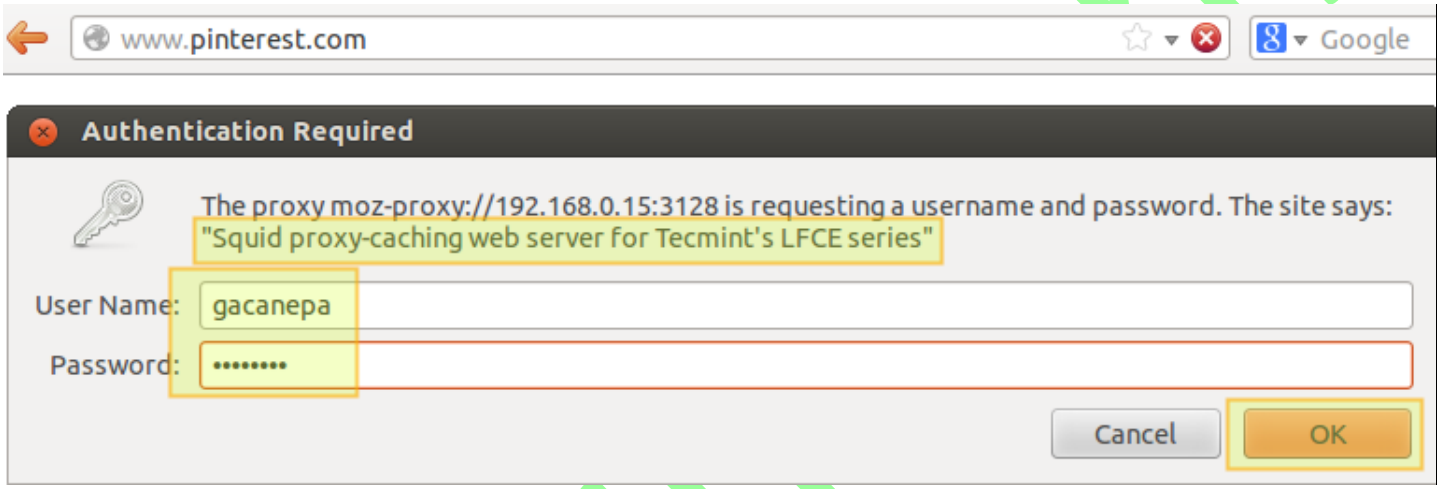
Run the following command to create the file and to add credentials for user gacanepa (omit the -c flag if the file already exists):

```
htpasswd -c /etc/squid/passwd gacanepa
```

```
root@dev2:~# htpasswd -c /etc/squid/passwd gacanepa
New password:
Re-type new password:
Adding password for user gacanepa
root@dev2:~# cat /etc/squid/passwd
gacanepa:$apr1$TbvBuvTq$p6kX3c/SpMgNp6wRmEVUM/
root@dev2:~#
```

By default, htpasswd will use MD5 encryption for passwords.

Open a web browser in the client machine and try to browse to any given site:



If authentication succeeds, access is granted to the requested resource. Otherwise, access will be denied.

Using cache to speed up data transfer

One of Squid's distinguishing features is the possibility of caching resources requested from the web to disk in order to speed up future requests of those objects either by the same client or others.

Add the following directives in your squid.conf file:

```
maximum_object_size 100 MB
cache_dir ufs /var/cache/squid 1000 16 256
refresh_pattern .*\.(mp4|iso) 2880
```

Where:

- ufs is the Squid storage format
- /var/cache/squid is a top-level directory where cache files will be stored. This directory must exist and be writable by Squid (Squid will NOT create this directory for you).
- 1000 is the amount (in MB) to use under this directory
- 16 is the number of 1st-level subdirectories, whereas 256 is the number of 2nd-level subdirectories within /var/spool/squid.
- The maximum_object_size directive specifies the maximum size of allowed objects in the cache.

- refresh_pattern tells Squid how to deal with specific file types (.mp4 and .iso in this case) and for how long it should store the requested objects in cache (2880 minutes = 2 days). The first and second 2880 are is an lower and upper limits, respectively, on how long objects without an explicit expiry time will be considered recent, and thus will be served by the cache, whereas 0% is the percentage of the objects' age (time since last modification) that each object without explicit expiry time will be considered recent.

Case study: downloading a .mp4 file from 2 different clients and testing the cache

First client (IP 192.168.0.104) downloads a 71 MB .mp4 file in 2 minutes and 52 seconds:

```
gacanepa@ubuntuOS:~$ wget http://media2.ldscdn.org/assets/first-presidencys-christmas-devotional/2014-firs
t-presidencys-christmas-devotional/2014-12-0010-elder-richard-j-maynes-360p-spa.mp4
--2014-12-08 14:06:46--  http://media2.ldscdn.org/assets/first-presidencys-christmas-devotional/2014-first
-presidencys-christmas-devotional/2014-12-0010-elder-richard-j-maynes-360p-spa.mp4
Connecting to 192.168.0.15:3128... connected.
Proxy request sent, awaiting response... 200 OK
Length: 74080266 (71M) [video/mp4]
Saving to: `2014-12-0010-elder-richard-j-maynes-360p-spa.mp4'

100%[======================================================================>] 74,080,266  447K/s   in 2m 52s

2014-12-08 14:09:38 (421 KB/s) - `2014-12-0010-elder-richard-j-maynes-360p-spa.mp4' saved [74080266/740802
66]

gacanepa@ubuntuOS:~$
```

Second client (IP 192.168.0.17) downloads the same file in 1.4 seconds!

```
[root@dev1 ~]# wget http://media2.ldscdn.org/assets/first-presidencys-christmas-devotional/2014-first-presi
dencys-christmas-devotional/2014-12-0010-elder-richard-j-maynes-360p-spa.mp4
--2014-12-08 14:11:34--  http://media2.ldscdn.org/assets/first-presidencys-christmas-devotional/2014-first-
presidencys-christmas-devotional/2014-12-0010-elder-richard-j-maynes-360p-spa.mp4
Connecting to 192.168.0.15:3128... connected.
Proxy request sent, awaiting response... 200 OK
Length: 74080266 (71M) [video/mp4]
Saving to: '2014-12-0010-elder-richard-j-maynes-360p-spa.mp4'

100%[======================================================================>] 74,080,266  51.2MB/s   in 1.4s

2014-12-08 14:11:36 (51.2 MB/s) - '2014-12-0010-elder-richard-j-maynes-360p-spa.mp4' saved [74080266/740802
66]

[root@dev1 ~]#
```

That is because the file was served from the Squid cache (indicated by TCP_HIT/200) in the second case, as opposed to the first instance, when it was downloaded directly from the Internet (represented by TCP_MISS/200). The HIT and MISS keywords, along with the 200 http response code, indicate that the file was served successfully both times, but the cache was HIT and MISSed respectively. When a request cannot be served by the cache for some reason, then Squid attempts to serve it from the Internet.

```
1418058581.197 172420 192.168.0.104 TCP_MISS/200 74080863 GET http://media2.ldscdn.org/assets/fi
ncys-christmas-devotional/2014-first-presidencys-christmas-devotional/2014-12-0010-elder-richard
60p-spa.mp4 - DIRECT/200.9.157.72 video/mp4
1418058696.106   1377 192.168.0.17 TCP_HIT/200 74080871 GET http://media2.ldscdn.org/assets/firs
ys-christmas-devotional/2014-first-presidencys-christmas-devotional/2014-12-0010-elder-richard-j
p-spa.mp4 - NONE/- video/mp4
```

Summary

In this chapter we have discussed how to set up a Squid web caching proxy. You can use the proxy server to filter contents using a chosen criteria, and also to reduce latency (since identical incoming requests are served from the cache, which is closer to the client than the web server that is actually serving the content, resulting in faster data transfers) and network traffic as well (reducing the amount of used bandwidth, which saves you money if you're paying for traffic).

You may want to refer to the Squid web site for further documentation (make sure to also check the wiki).

Chapter 25: SquidGuard

In this chapter we will explain how to use squidGuard, a filter, redirector and access controller plugin for squid. Let's start our discussion by highlighting what squidGuard can and cannot do:

squidGuard can be used to:

- limit the allowed web access for some users to a list of accepted/well known web servers and/or URLs only, while denying access to other blacklisted web servers and/or URLs.
- block access to sites (by IP address or domain name) matching a list of regular expressions or words for some users.
- require the use of domain names/prohibit the use of IP address in URLs.
- redirect blocked URLs to error or info pages.
- use distinct access rules based on time of day, day of the week, date etc.
- implement different rules for distinct user groups.

However, neither squidGuard nor Squid can be used to

- analyze text inside documents and act in result.
- detect or block embedded scripting languages like JavaScript, Python, or VBscript inside HTML code.

Blacklists – the basics

Blacklists are an essential part of squidGuard. Basically, they are plain text files that will allow you to implement content filters based on specific keywords. There are both freely available and commercial blacklists, and you can find the download links in the project's website.

In this tutorial I will show you how to integrate the blacklists provided by Shalla Secure Services (http://www.shallalist.de/) to your squidGuard installation. These blacklists are free for personal / non-commercial use and are updated on a daily basis. They include, as of today, over 1,700,000 entries.

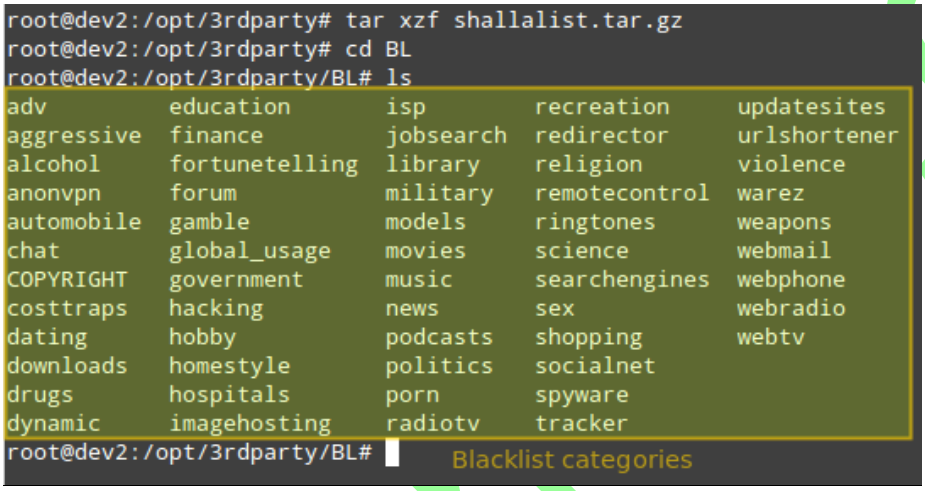
For our convenience, let's create a directory to download the blacklist package:

```
mkdir /opt/3rdparty
cd /opt/3rdparty
wget http://www.shallalist.de/Downloads/shallalist.tar.gz
```

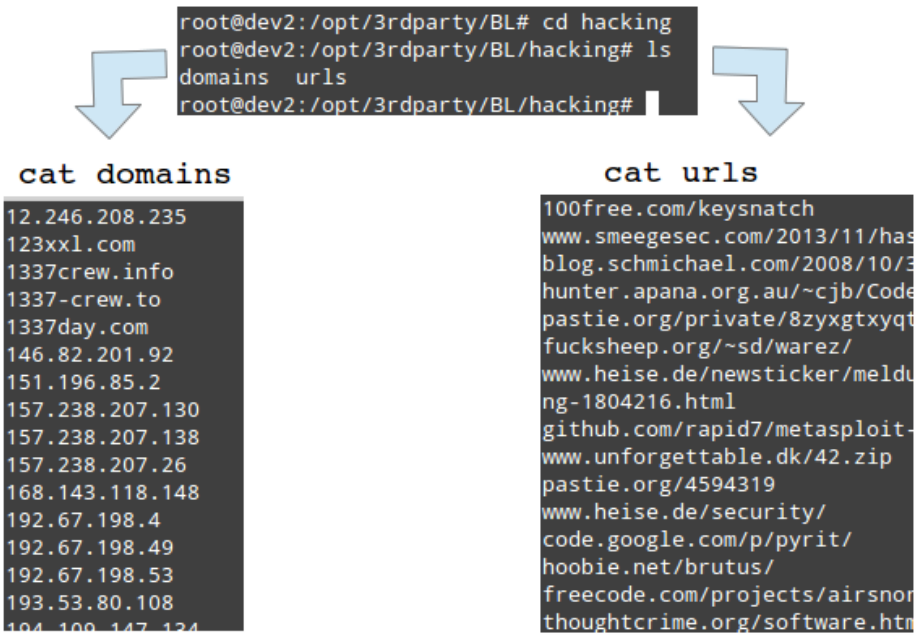
The latest download link is always available as highlighted below:

After untarring the newly downloaded file, we will browse to the blacklist (BL) folder:

```
tar xzf shallalist.tar.gz
cd BL
ls
```



You can think of the directories shown in the output of ls as backlist categories, and their corresponding (optional) subdirectories as subcategories, descending all the way down to specific URLs and domains, which are listed in the files urls and domains, respectively. Refer to the below image for further details:



Installing blacklists

Installation of the whole blacklist package, or of individual categories, is performed by copying the BL directory, or one of its subdirectories, respectively, to the /var/squidGuard/db directory. Of course you could have downloaded the blacklist tarball to this directory in the first place, but the approach explained earlier gives you more control over what categories should be blocked (or not) at a specific time.

Next, I will show you how to install the anonvpn, hacking, and chat blacklists and how to configure squidGuard to use them.

Please note that this chapter was written using CentOS 7. If you are using another distribution, the squidGuard database should be located in a similar directory under /var.

Step 1: Copy recursively the anonvpn, hacking, and chat directories from /opt/3rdparty/BL to /var/squidGuard/db

```
cp -a /opt/3rdparty/BL/anonvpn /var/squidGuard/db
cp -a /opt/3rdparty/BL/hacking /var/squidGuard/db
cp -a /opt/3rdparty/BL/chat /var/squidGuard/db
```

Step 2: Use the domains and urls files to create squidguard's database files. Please note that the following command will work for creating .db files for all the installed blacklists - even when a certain category has 2 or more subcategories.

```
squidGuard -d -C all
```

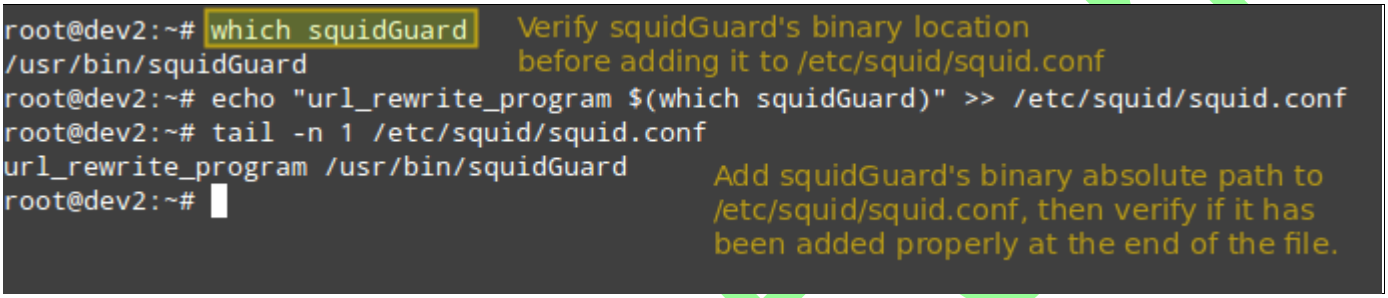
Step 3: Change the ownership of the /var/squidGuard/db/ directory and its contents to the proxy user so that Squid can read the database files

```
chown -R proxy:proxy /var/squidGuard/db/
```

Step 4: Configure Squid to use squidGuard

We will use Squid's url_rewrite_program directive in /etc/squid/squid.conf to tell Squid to use squidGuard as a URL rewriter / redirector. Add the following line to squid.conf, making sure that /usr/bin/squidGuard is the right absolute path in your case:

```
which squidGuard
echo "url_rewrite_program $(which squidGuard)" >> /etc/squid/squid.conf
tail -n 1 /etc/squid/squid.conf
```

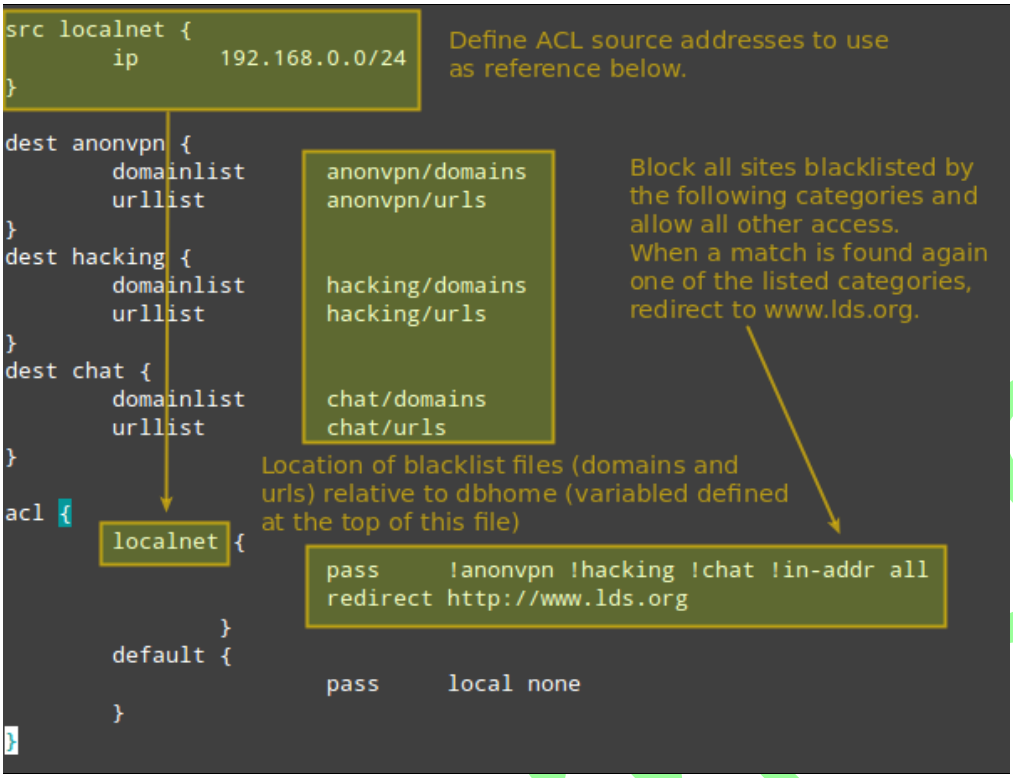


Step 5: Add the necessary directives to squidGuard's configuration file (located in /etc/squidguard/squidGuard.conf)

Please refer to the screenshot after the following code for further clarification

```
src localnet {
        ip      192.168.0.0/24
}
dest anonvpn {
        domainlist      anonvpn/domains
        urllist         anonvpn/urls
}
dest hacking {
        domainlist      hacking/domains
        urllist         hacking/urls
}
dest chat {
        domainlist      chat/domains
        urllist         chat/urls
}
acl {
        localnet {
                        pass     !anonvpn !hacking !chat !in-addr all
                        redirect http://www.lds.org
                }
        default {
                        pass     local none
```

```
        }
}
```



Step 6: Restart Squid and test

```
service squid restart # sysvinit / Upstart-based systems
systemctl restart squid.service # systemctl-based systems
```

Open a web browser in a client within local network and browse to a site found in any of the blacklist files (domains or urls - we will use http://spin.de/chat in the following example) and you will be redirected to another URL, www.lds.org in this case.

You can verify that the request was made to the proxy server but was denied (301 http response - Moved permanently) and was redirected to www.lds.org instead:

```
418434849.491 115916 192.168.0.103 TCP_MISS/200 2163 CONNECT edge.ldscdn.org:443 - DIRECT/23.12.151.117 -
418434852.492 116168 192.168.0.103 TCP_MISS/200 2816 CONNECT www.lds.org:443 - DIRECT/23.12.151.116 -
418434866.767    409 192.168.0.103 TCP_MISS/301 371 GET http://spin.de/chat - DIRECT/23.12.151.116 -
418434881.309  11857 192.168.0.103 TCP_MISS/200 5063 CONNECT ldschurch.tt.omtrdc.net:443 - DIRECT/66.235.132.179 -
418434881.417  13863 192.168.0.103 TCP_MISS/200 8837 CONNECT ldschurch.tt.omtrdc.net:443 - DIRECT/66.235.132.173 -
418434885.820  16367 192.168.0.103 TCP_MISS/200 2892 CONNECT om.lds.org:443 - DIRECT/63.140.59.9 -
```

Removing restrictions

If for some reason you need to enable a category that has been blocked in the past, remove the corresponding directory from /var/squidGuard/db and comment (or delete) the related acl in the squidguard.conf file.

For example, if you want to enable the domains and urls blacklisted by the anonvpn category, you would need to perform the following steps:

```
rm -rf /var/squidGuard/db/anonvpn
```

And edit the squidguard.conf file as follows:

BEFORE

```
src localnet {
        ip      192.168.0.0/24
}

dest anonvpn {
        domainlist      anonvpn/domains
        urllist         anonvpn/urls
}
dest hacking {
        domainlist      hacking/domains
        urllist         hacking/urls
}
dest chat {
        domainlist      chat/domains
        urllist         chat/urls
}

acl {
        localnet {
                        pass     !anonvpn !hacking !chat !in-addr all
                        redirect http://www.lds.org
                }
        default {
                        pass     local none
        }
}
```

AFTER

```
src localnet {
        ip      192.168.0.0/24
}

dest hacking {
        domainlist      hacking/domains
        urllist         hacking/urls
}
dest chat {
        domainlist      chat/domains
        urllist         chat/urls
}

acl {
        localnet {
                        pass     !hacking !chat !in-addr all
                        redirect http://www.lds.org
                }
        default {
                        pass     local none
        }
}
```

Please note that parts highlighted in yellow under BEFORE have been deleted in AFTER doing:

```
squidGuard –d -C all
squid –k reconfigure
```

Whitelisting specific domains and URLs

On occasions you may want to allow certain URLs or domains, but not an entire blacklisted directory. In that case, you should create a directory named myWhiteLists (or whatever name you choose) and insert the desired URLs and domains under /var/squidGuard/db/myWhiteLists in files named urls and domains, respectively.

Then, initialize the new content rules as before,

```
squidGuard -C all
```

and modify the squidguard.conf as follows:

BEFORE

```
src localnet {
        ip      192.168.0.0/24
}

dest anonvpn {
        domainlist      anonvpn/domains
        urllist         anonvpn/urls
}
dest hacking {
        domainlist      hacking/domains
        urllist         hacking/urls
}
dest chat {
        domainlist      chat/domains
        urllist         chat/urls
}

acl {
        localnet {
                        pass     !anonvpn !hacking !chat !in-addr all
                        redirect http://www.lds.org
                }
        default {
                        pass     local none
        }
}
```

AFTER

```
src localnet {
        ip      192.168.0.0/24
}

dest myWhiteLists {
        domainlist      myWhiteLists/domains
        urllist         myWhiteLists/urls
}

dest anonvpn {
        domainlist      anonvpn/domains
        urllist         anonvpn/urls
}
dest hacking {
        domainlist      hacking/domains
        urllist         hacking/urls
}
dest chat {
        domainlist      chat/domains
        urllist         chat/urls
}

acl {
        localnet {
                  pass myWhiteLists !anonvpn !hacking !chat !in-addr
all
                        redirect http://www.lds.org
                }
        default {
                        pass     local none
        }
}
```

As before, the parts highlighted in yellow indicate the changes that need to be added. Note that the myWhiteLists string needs to be first in the row that starts with pass.

Finally, remember to restart Squid in order to apply changes.

Summary

After following the steps outlined in this tutorial you should have a powerful content filter and URL redirector working hand in hand with your Squid proxy. If you experience any issues during your installation / configuration process or have any questions or comments, you may want to refer to squidGuard's web documentation.

Chapter 26: Mail server

Regardless of the many online communication methods that are available today, email remains a practical way to deliver messages from one end of the world to another, or to a person sitting in the office next to ours.

In this chapter we will explain how to configure your mail server and how to perform the following tasks:

-Configure email aliases

-Configure an IMAP and IMAPS service

-Configure an smtp service

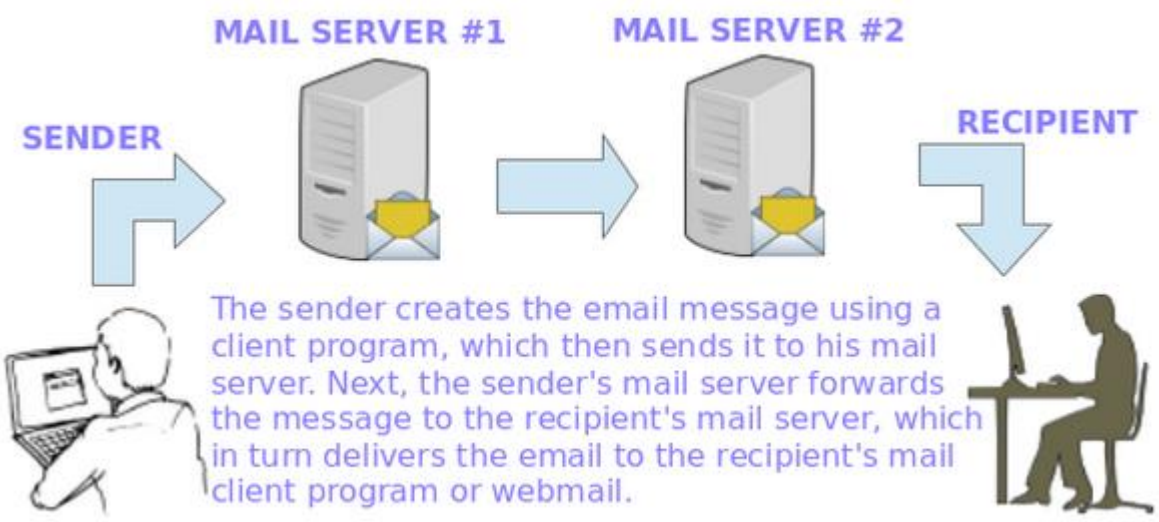
-Restrict access to an smtp server

Note that our setup will only cover a mail server for a local area network where the machines belong to the same domain. Sending email messages to other domains require a more complex setup, including domain name resolution capabilities, that is out of the scope of the LFCE certification.

But first off, let's start with a few definitions.

The process of sending and receiving email messages

The following image illustrates the process of email transport starting with the sender until the message reaches the recipient's inbox:



To make this possible, several things happen behind the scenes. In order for an email message to be delivered from a client application (such as Thunderbird, Outlook, or webmail services such as Gmail or Yahoo! Mail) to his / her mail server and from there to the destination server and finally to its intended recipient, a SMTP (Simple Mail Transfer Protocol) service must be in place in each server.

In order for these components to be able to "talk" to each other, they must "speak" the same "language" (or protocol), namely SMTP as defined in the RFC 2821. Most likely, you will have to refer to that RFC while setting up your mail server environment.

Other protocols that we need to take into account are IMAP (Internet Message Access Protocol), which allows to manage email messages directly on the server without downloading them to our client's hard drive, and POP3 (Post Office Protocol), which allows to download the messages and folders to the user's computer.

Our testing environment

Our testing environment is as follows:

- Mail server: Debian Wheezy 7.5 [IP 192.168.0.15]
- Mail client: Ubuntu 12.04 [IP 192.168.0.103]
- Local domain: example.com.ar
- Aliases: sysadmin@example.com.ar is aliased to gacanepa@example.com.ar and jdoe@example.com.ar
- On our client, we have set up elementary DNS resolution adding the following line to the /etc/hosts file:

```
192.168.0.15 example.com.ar mailserver
```

Adding email aliases

By default, a message sent to a specific user should be delivered to that user only. However, if you want to also deliver it to a group of users as well, or to a different user, you can create a mail alias or use one of the existing ones in /etc/postfix/aliases, following this syntax:

```
user1: user1, user2
```

Thus, emails sent to user1 will be also delivered to user2. Note that if you omit the word user1 after the colon, as in

```
user1: user2
```

the messages sent to user1 will only be sent to user2, and not to user1.

In the above example, user1 and user2 should already exist on the system.

In our specific case, we will use the following alias as explained before (add the following line in /etc/aliases):

```
sysadmin: gacanepa, jdoe
```

and run

```
postalias /etc/postfix/aliases
```

to create or refresh the aliases lookup table. Thus, messages sent to sysadmin@example.com.ar will be delivered to the inbox of the users listed above.

Configuring Postfix – the SMTP service

The main configuration file for Postfix is /etc/postfix/main.cf. You only need to set up a few parameters before being able to use the mail service. However, you should become acquainted with the full configuration parameters (which can be listed with man 5 postconf) in order to set up a secure and fully customized mail server. Note that this tutorial is only supposed to get you started in that process and does not represent a comprehensive guide on email services with Linux.

1) **myorigin** specifies the domain that appears in messages sent from the server. You may see the /etc/mailname file used with this parameter. Feel free to edit it if needed.

```
myorigin = /etc/mailname
```

```
root@dev2:~# cd /etc/postfix
root@dev2:/etc/postfix# grep myorigin main.cf
myorigin = /etc/mailname
root@dev2:/etc/postfix# cat /etc/mailname
example.com.ar
root@dev2:/etc/postfix#
```

If the value above is used, mails will be sent as user@debian.gabrielcanepa.com.ar, where user is the user sending the message.

2) **mydestination** lists what domains this machine will deliver email messages locally, instead of forwarding to another machine (acting as a relay system). The default settings will suffice in our case (make sure to edit the file to suit your environment):

```
root@dev2:/etc/postfix# grep mydestination main.cf
mydestination = debian.gabrielcanepa.com.ar, localhost.gabrielcanepa.com.ar, localhost, hash:/et
relay_domains = $mydestination
root@dev2:/etc/postfix#
```

where the /etc/postfix/transport file defines the relationship between domains and the next server to which mail messages should be forwarded. In our case, since we will be delivering messages to our local area network only (thus bypassing any external DNS resolution), the following configuration will suffice:

```
example.com.ar    local:
.example.com.ar    local:
```

Next, we need to convert this plain text file to the .db format, which creates the lookup table that Postfix will actually use to know what to do with incoming and outgoing mail:

```
postmap /etc/postfix/transport
```

You will need to remember to recreate this table if you add more entries to the corresponding text file.

3) **mynetworks** defines the authorized networks Postfix will forward messages from. The default value, subnet, tells Postfix to forward mail from SMTP clients in the same IP subnetworks as the local machine only.

```
mynetworks = subnet
```

```
root@dev2:/etc/postfix# grep mynetworks main.cf
mynetworks = subnet
root@dev2:/etc/postfix#
```

4) **relay_domains** specifies the destinations to which emails should be sent to. We will leave the default value untouched, which points to mydestination. Remember that we are setting up a mail server for our LAN.

```
relay_domains = $mydestination
```

Note that you can use $mydestination instead of listing the actual contents.

```
root@dev2:/etc/postfix# grep relay_domains main.cf
relay_domains = $mydestination
root@dev2:/etc/postfix#
```

Next,

5) **inet_interfaces** defines which network interfaces the mail service should listen on. The default, all, tells Postfix to use all network interfaces.

```
inet_interfaces = all
```

```
root@dev2:/etc/postfix# grep inet main.cf
inet_interfaces = all
root@dev2:/etc/postfix#
```

Finally,

6) **mailbox_size_limit** and **message_size_limit** will be used to set the size of each user's mailbox and the maximum allowed size of individual messages, respectively, in bytes.

```
mailbox_size_limit = 51200000
message_size_limit = 5120000
```

Restricting access to the SMTP server

The Postfix SMTP server can apply certain restrictions to each client connection request. Not all clients should be allowed to identify themselves to the mail server using the smtp HELO command, and certainly not all of them should be granted access to send or receive messages.

To implement these restrictions, we will use the following directives in the main.cf file. Though they are self-explanatory, comments have been added for clarification purposes:

```
# Require that a remote SMTP client introduces itself with the HELO or EHLO
command before sending the MAIL command or other commands that require EHLO
negotiation.
smtpd_helo_required = yes
# Permit the request when the client IP address matches any network or network
address listed in $mynetworks
# Reject the request when the client HELO and EHLO command has a bad hostname
syntax
smtpd_helo_restrictions = permit_mynetworks, reject_invalid_helo_hostname
# Reject the request when Postfix does not represent the final destination for
the sender address
smtpd_sender_restrictions = permit_mynetworks, reject_unknown_sender_domain
# Reject the request unless 1) Postfix is acting as mail forwarder or 2) is the
final destination
smtpd_recipient_restrictions = permit_mynetworks, reject_unauth_destination
```

The Postfix configuration parameters page may come in handy in order to further explore the available options.

Configuring Dovecot

Right after installing dovecot, it supports out-of-the-box the POP3 and IMAP protocols, along with their secure versions, POP3S and IMAPS, respectively.

Add the following lines in /etc/dovecot/conf.d/10-mail.conf:

```
# %u represents the user account that logs in
# Mailboxes are in mbox format
mail_location = mbox:~/mail:INBOX=/var/mail/%u
# Directory owned by the mail group and the directory set to group-writable
(mode=0770, group=mail)
# You may need to change this setting if postfix is running a different user /
group on your system
mail_privileged_group = mail
```

If you check your home directory, you will notice there is a mail subdirectory with the following contents:

```
root@dev2:/etc/dovecot# ls -l /home/gacanepa/mail
total 8
-rw------- 1 gacanepa gacanepa 1495 Dec 22 17:22 Drafts
-rw------- 1 gacanepa gacanepa 1362 Dec 22 17:38 Sent
-rw------- 1 gacanepa gacanepa    0 Dec 22 17:08 Trash
root@dev2:/etc/dovecot#
```

Also, please note that the /var/mail/%u file is where the user's mails are store on most systems.

Add the following directive to /etc/dovecot/dovecot.conf (note that imap and pop3 imply imaps and pop3s as well):

```
protocols = imap pop3
```

And make sure /etc/dovecot/conf.d/10-ssl.conf includes the following lines (otherwise, add them):

```
ssl_cert = </etc/dovecot/dovecot.pem
ssl_key = </etc/dovecot/private/dovecot.pem
```

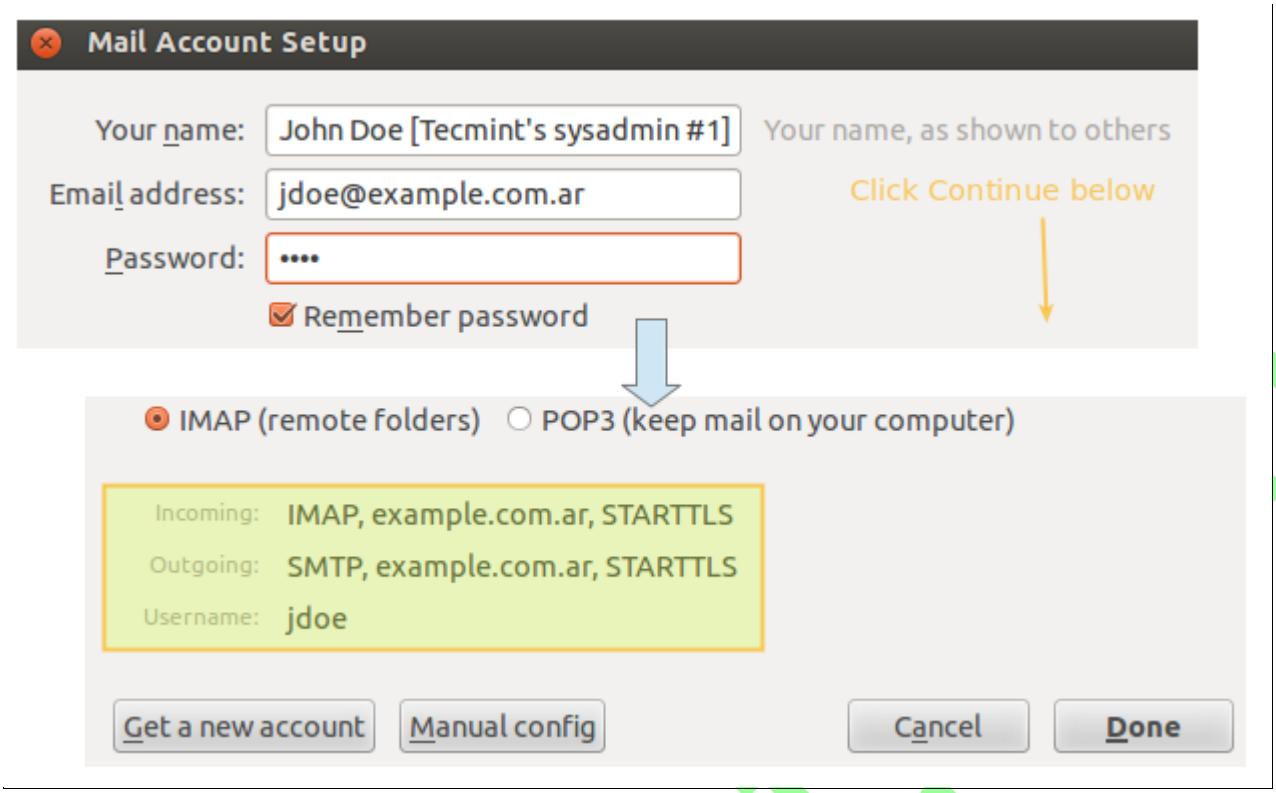
Now let's restart Dovecot and verify that it listens on the ports related to imap, imaps, pop3, and pop3s:

```
netstat -npltu | grep dovecot
```

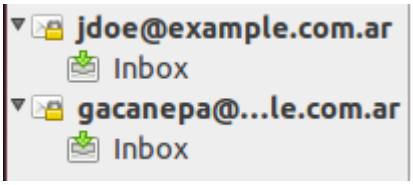
```
root@dev2:/etc/dovecot# netstat -npltu | grep dovecot
tcp        0      0 0.0.0.0:110             0.0.0.0:*               LISTEN      3810/dovecot
tcp        0      0 0.0.0.0:143             0.0.0.0:*               LISTEN      3810/dovecot
tcp        0      0 0.0.0.0:993             0.0.0.0:*               LISTEN      3810/dovecot
tcp        0      0 0.0.0.0:995             0.0.0.0:*               LISTEN      3810/dovecot
tcp6       0      0 :::110                  :::*                    LISTEN      3810/dovecot
tcp6       0      0 :::143                  :::*                    LISTEN      3810/dovecot
tcp6       0      0 :::993                  :::*                    LISTEN      3810/dovecot
tcp6       0      0 :::995                  :::*                    LISTEN      3810/dovecot
root@dev2:/etc/dovecot#
Port 110: POP3
Port 143: IMAP
Port 993: IMAP4 over SSL (IMAPS)
Port 995: Secure POP3 (SSL-POP)
```

Setting up a mail client and sending / receiving emails

On our client computer, we will open Thunderbird and click on File → New → Existing mail account. We will be prompted to enter the name of the account and the associated email address, along with its password. When we click Continue, Thunderbird will then try to connect to the mail server in order to verify settings:



Repeat the process above for the next account (gacanepa@example.com.ar) and the following two inboxes should appear in Thunderbird's left pane:



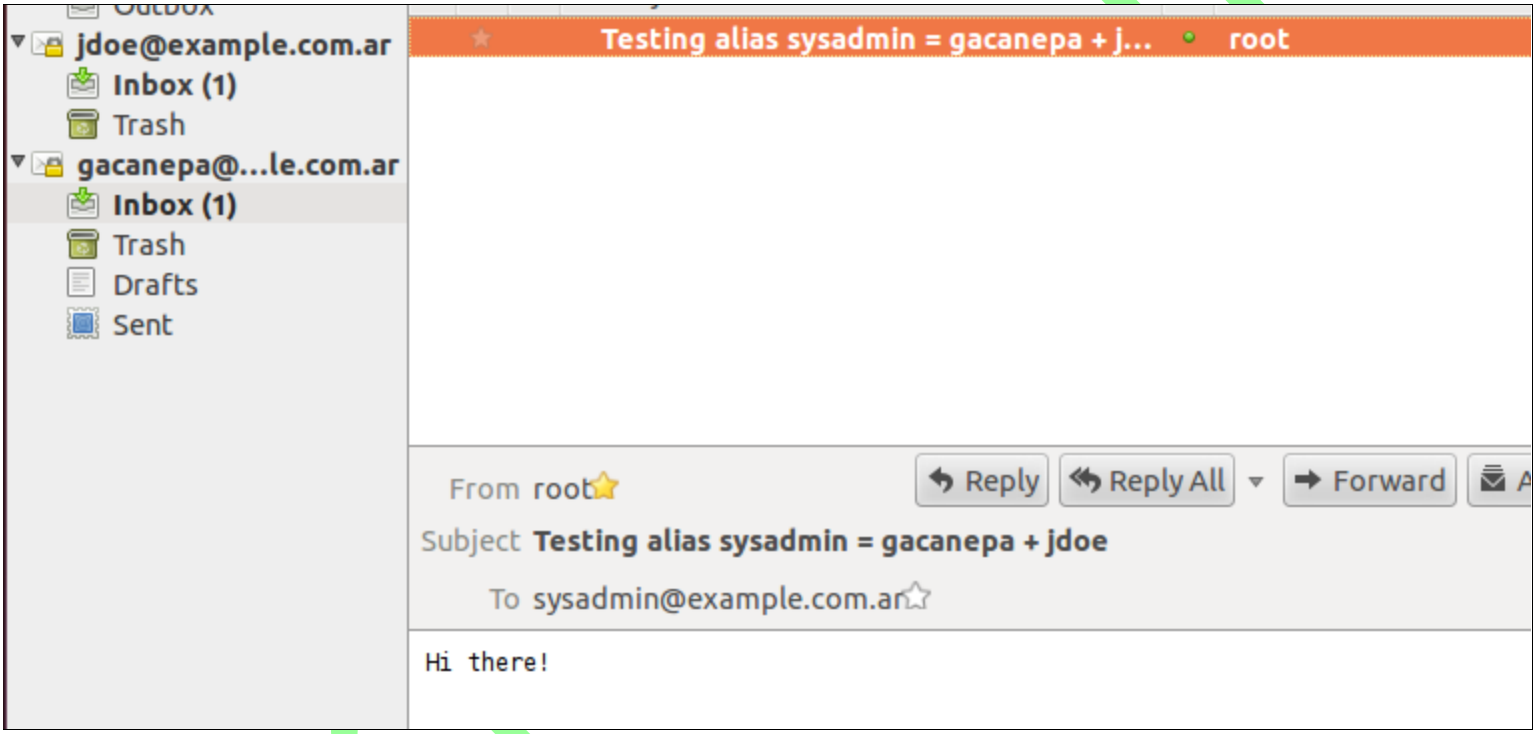
On our server, we will write an email message to sysadmin, which is aliased to jdoe and gacanepa:

```
root@dev2:~# mail sysadmin
Subject: Testing alias sysadmin = jdoe + gacanepa
Hi there!
.
Cc:
root@dev2:~#
```

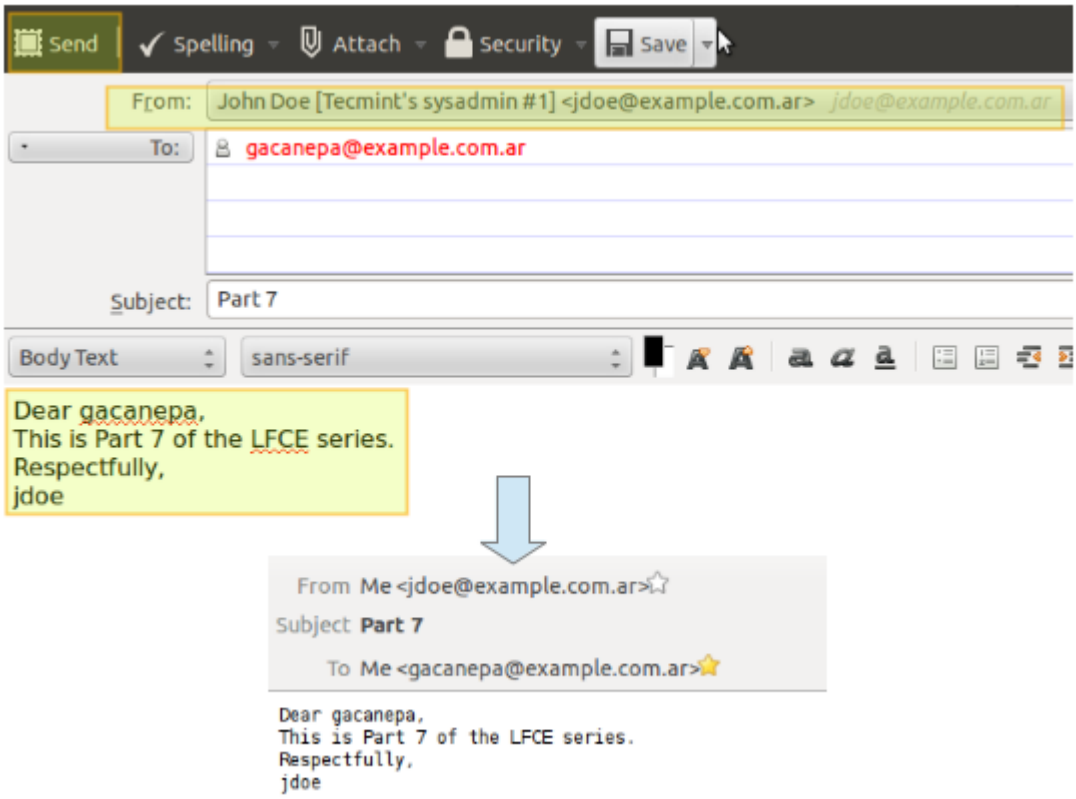
The mail log (/var/log/mail.log) seems to indicate that the email that was sent to sysadmin was relayed to jdoe@example.com.ar and gacanepa@example.com.ar, as can be seen in the following image:

```
Dec 22 23:49:38 dev2 postfix/pickup[4332]: 340666914: uid=0 from=<root>
Dec 22 23:49:38 dev2 postfix/cleanup[4842]: 340666914: message-id=<20141223024938.340666914@mail
Dec 22 23:49:38 dev2 postfix/qmgr[4333]: 340666914: from=<root@example.com.ar>, size=345, nrcpt=
Dec 22 23:49:38 dev2 postfix/local[4844]: 340666914: to=<gacanepa@example.com.ar>, orig_to=<sysa
s=sent (delivered to command: procmail -a "$EXTENSION")
Dec 22 23:49:38 dev2 postfix/local[4844]: 340666914: to=<jdoe@example.com.ar>, orig_to=<sysadmin
ent (delivered to command: procmail -a "$EXTENSION")
Dec 22 23:49:38 dev2 postfix/qmgr[4333]: 340666914: removed
```

We can verify if the mail was actually delivered to our client, where the IMAP accounts were configured in Thunderbird:



Finally, let's try to send a message from jdoe@example.com.ar to gacanepa@example.com.ar:



In the exam you will be asked to work exclusively with command-line utilities. This means you will not be able to install a desktop client application such as Thunderbird, but will be required to use **mail** instead. We have used Thunderbird in this chapter for illustrative purposes only.

Summary

In this post we have explained how to set up an IMAP mail server for your local area network and how to restrict access to the SMTP server. If you happen to run into an issue while implementing a similar setup in your testing environment, you will want to check the online documentation of Postfix and Dovecot (specially the pages about the main configuration files, /etc/postfix/main.cf and /etc/dovecot/dovecot.conf, respectively).

Chapter 27: The firewall

You will recall from Chapter 18 ("Network services") that we gave a basic description of what a firewall is: a mechanism to manage packets coming into and leaving the network. By "manage" we actually mean

1) to allow or prevent certain packages to enter or leave our network.

2) to forward other packages from one point of the network to another.

based on predetermined criteria.

In this chapter we will discuss how to implement basic packet filtering and how to configure the firewall with **iptables**, a frontend to **netfilter**, which is a native kernel module used for firewalling.

Please note that firewalling is a vast subject and this article is not intended to be a comprehensive guide to understanding all that there is to know about it, but rather as a starting point for a deeper study of this topic.

You can think of a firewall as an international airport where passenger planes come and go almost 24/7. Based on a number of conditions, such as the validity of a person's passport, or his / her country of origin (to name a few examples) he or she may, or may not, be allowed to enter or leave a certain country. At the same time, airport officers can instruct people to move from one place of the airport to another if necessary, for example when they need to go through Customs Services.

We may find the airport analogy useful during the rest of this tutorial. Just keep in mind the following relations as we proceed:

- Persons = Packets
- Firewall = Airport
- Country #1 = Network #1
- Country #2 = Network #2
- Airport regulations enforced by officers = firewall rules

The basics about iptables

At the low level, it is the kernel itself which "decides" what to do with packages based on rules grouped in chains, or sentences. These chains define what actions should be taken when a package matches the criteria specified by them.

The first action taken by iptables will consist in deciding what to do with a packet:

- Accept it (let it go through into our network)?
- Reject it (prevent it from accessing our network)?
- Forward it (to another chain)?

Just in case you were wondering why this tool is called iptables, it's because these chains are organized in tables, with the filter table being the most well know and the one that is used to implement packet filtering with its three default chains:

1) The **INPUT** chain handles packets coming into the network, which are destined for local programs.

2) The **OUTPUT** chain is used to analyze packets originated in the local network, which are to be sent to the outside.

Finally,

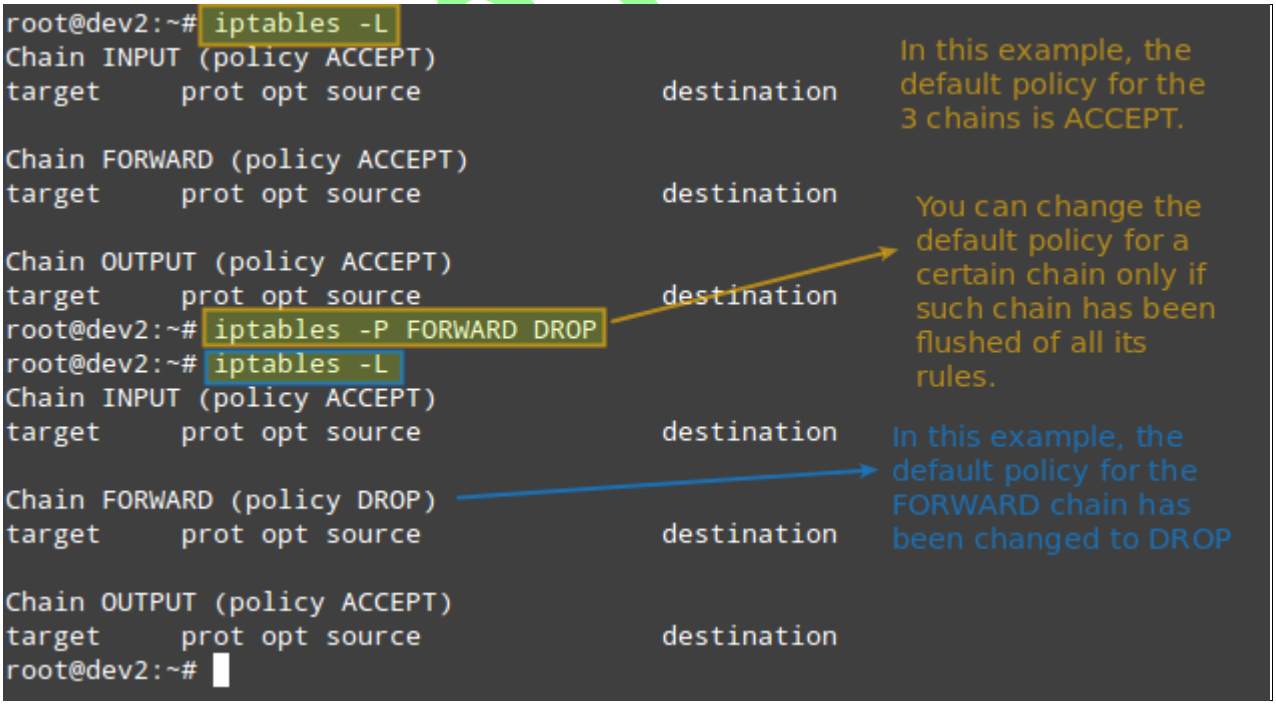
3) The **FORWARD** chain processes the packets which should be forwarded to another destination (as in the case of a router).

For each of these chains there is a default policy, which dictates what should be done by default when packets do not match any of the rules in the chain. You can view the rules created for each chain and the default policy by running the following command:

```
iptables -L
```

The available policies are as follows:

- ACCEPT → lets the packet through. Any packet that does not match any rules in the chain is allowed into the network.
- DROP → drops the packet quietly. Any packet that does not match any rules in the chain is prevented from entering the network.
- REJECT → rejects the packet and returns informative message. This one in particular does not work as a default policy. Instead, it is meant to complement packet filtering rules.



When it comes to deciding which policy you will implement, you need to consider the pros and cons of each approach as explained above - note that there is no one-size-fits-all solution.

Adding rules

To add a rule to the firewall, invoke the iptables command as follows:

```
iptables -A chain_name criteria -j target
```

where

- -A stands for Append (append the current rule to the end of the chain)
- chain_name is either INPUT, OUTPUT, or FORWARD.
- target is the action, or policy, to apply in this case (ACCEPT, REJECT, or DROP).
- criteria is the set of conditions against which the packets are to be examined. It is composed of at least one (most likely more) of the following flags. Options inside brackets, separated by a vertical bar, are equivalent to each other. The rest represents optional switches:

```
[--protocol | -p] protocol: specifies the protocol involved in a rule.
[--source-port | -sport] port:[port]: defines the port (or range of ports) where
the packet originated.
[--destination-port | -dport] port:[port]: defines the port (or range of ports)
to which the packet is destined.
[--source | -s] address[/mask]: represents the source address or network/mask.
[--destination | -d] address[/mask]: represents the destination address or
network/mask.
[--state] state (preceded by -m state): manage packets depending on whether they
are part of a state connection, where state can be NEW, ESTABLISHED, RELATED, or
INVALID.
[--in-interface | -i] interface: specifies the input interface of the packet.
[--out-interface | -o] interface: the output interface.
[--jump | -j] target: what to do when the packet matches the rule.
```

Let’s glue all that in 3 classic examples using the following test environment for the first two:

- Firewall: Debian Wheezy 7.5 [dev2 → 192.168.0.15]
- Source: CentOS 7 [dev1 → 192.168.0.17]

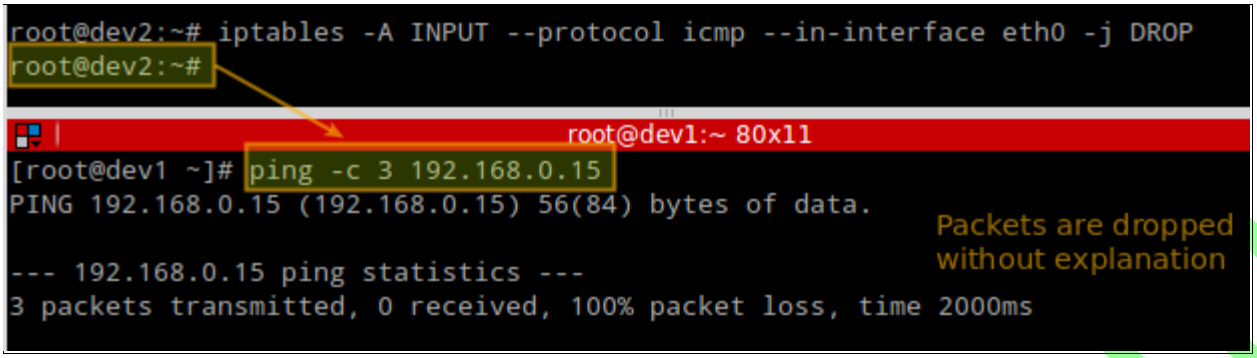
And this for the last example

- NFSv4 server and firewall: Debian Wheezy 7.5 [debian → 192.168.0.10]
- Source: Debian Wheezy 7.5 [dev2 → 192.168.0.15]

Example 1: Analyzing the difference between the DROP and REJECT policies

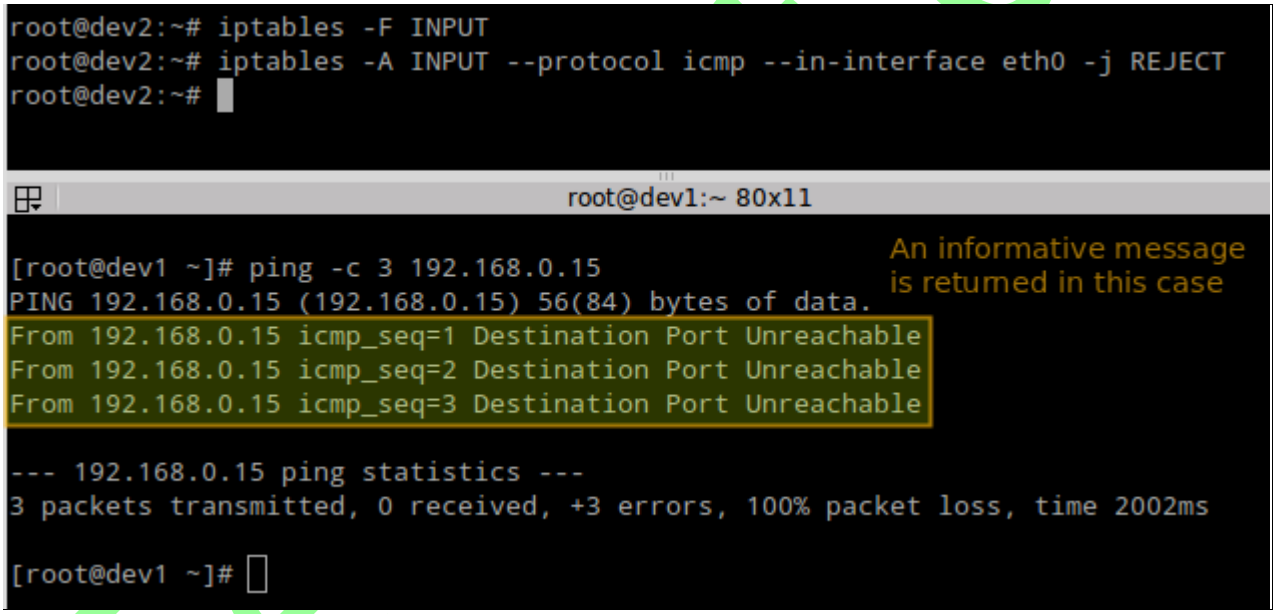
We will define a DROP policy first for input pings to our firewall. That is, icmp packets will be dropped quietly.

```
ping -c 3 192.168.0.15
iptables -A INPUT --protocol icmp --in-interface eth0 -j DROP
```



Before proceeding with the REJECT part, we will flush all rules from the INPUT chain to make sure our packets will be tested by this new rule:

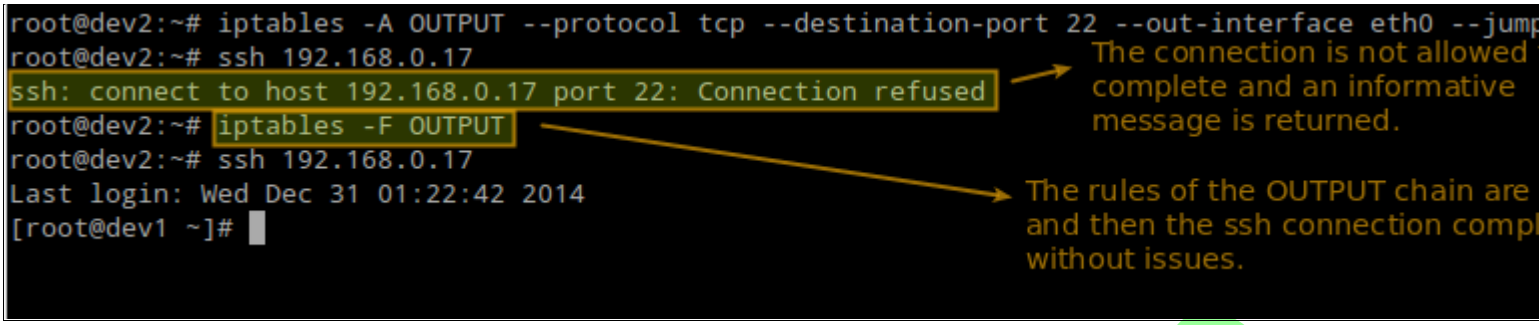
```
iptables -F INPUT
iptables -A INPUT --protocol icmp --in-interface eth0 -j REJECT
```



Example 2: Disabling / re-enabling ssh logins from dev2 to dev1

We will be dealing with the OUTPUT chain as we're handling outgoing traffic:

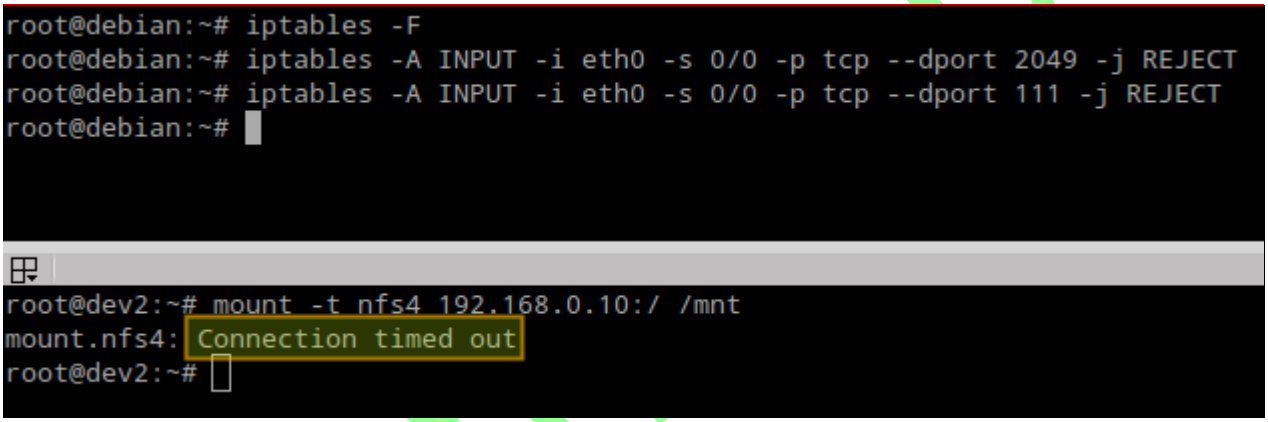
```
iptables -A OUTPUT --protocol tcp --destination-port 22 --out-interface eth0 --
jump REJECT
```



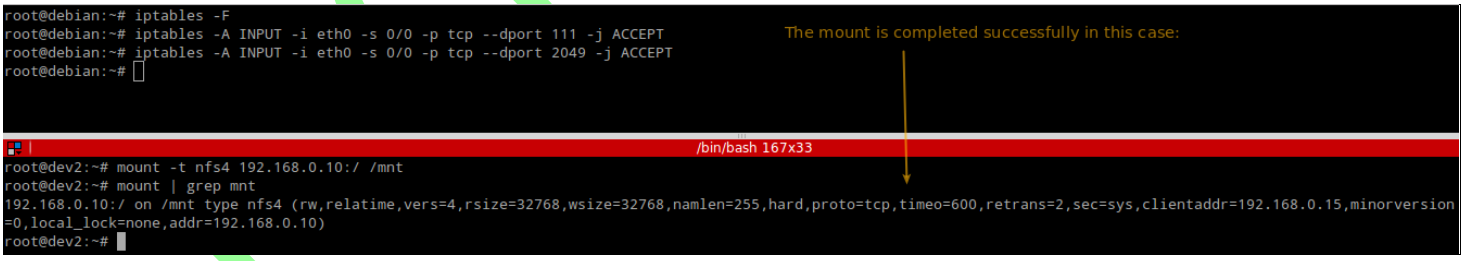
Example 3: Allowing / preventing NFS clients (from 192.168.0.0/24) to mount NFS4 shares

Run the following commands in the NFSv4 server / firewall to close ports 2049 and 111 for all kind of traffic:

```
iptables -F
iptables -A INPUT -i eth0 -s 0/0 -p tcp --dport 2049 -j REJECT
iptables -A INPUT -i eth0 -s 0/0 -p tcp --dport 111 -j REJECT
```



Now let's open those ports and see what happens:



As you can see, we were able to mount the NFSv4 share after opening the traffic.

Inserting, appending, and deleting rules

In the previous examples we showed how to append rules to the INPUT and OUTPUT chains. Should we want to insert them instead at a predefined position, we should use the -I (uppercase i) switch instead.

You need to remember that rules will be evaluated one after another, and that the evaluation stops (or jumps) when a DROP or ACCEPT policy is matched. For that reason, you may find yourself in the need to move rules up or down in the chain list as needed.

We will use a trivial example to demonstrate this:

```
root@debian:~# iptables -nL -v --line-numbers
Chain INPUT (policy ACCEPT 167 packets, 12556 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 ACCEPT     tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:111
2       16  1912 ACCEPT     tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:2049

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 110 packets, 12536 bytes)
num   pkts bytes target     prot opt in     out     source               destination
```

Let's place the following rule

```
iptables -I INPUT 2 -p tcp --dport 80 -j ACCEPT
```

at position 2) in the INPUT chain (thus moving previous #2 as #3)

```
root@debian:~# iptables -nL -v --line-numbers
Chain INPUT (policy ACCEPT 13 packets, 964 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        0     0 ACCEPT     tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:111
2        0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:80
3       16  1912 ACCEPT     tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/0           tcp dpt:2049

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 7 packets, 1100 bytes)
num   pkts bytes target     prot opt in     out     source               destination
```

Using the setup above, traffic will be checked to see whether it's directed to port 80 before checking for port 2049.

Alternatively, you can delete a rule and change the target of the remaining rules to DROP (using the -R switch):

```
iptables -D INPUT 1
iptables -nL -v --line-numbers
iptables -R INPUT 2 -i eth0 -s 0/0 -p tcp --dport 2049 -j REJECT
iptables -R INPUT 1 -p tcp --dport 80 -j REJECT
```

```
root@debian:~# iptables -D INPUT 1
root@debian:~# iptables -nL -v --line-numbers
Chain INPUT (policy ACCEPT 213 packets, 17548 bytes)
num   pkts bytes target     prot opt in     out     source               destinat
1        0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/
2       22  2348 ACCEPT     tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destinat

Chain OUTPUT (policy ACCEPT 128 packets, 12784 bytes)
num   pkts bytes target     prot opt in     out     source               destinat
root@debian:~# iptables -R INPUT 2 -i eth0 -s 0/0 -p tcp --dport 2049 -j REJECT
root@debian:~# iptables -R INPUT 1 -p tcp --dport 80 -j REJECT
root@debian:~# iptables -nL -v --line-numbers
Chain INPUT (policy ACCEPT 9 packets, 660 bytes)
num   pkts bytes target     prot opt in     out     source               destinat
1        0     0 REJECT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/
2        0     0 REJECT     tcp  --  eth0   *       0.0.0.0/0            0.0.0.0/

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destinat

Chain OUTPUT (policy ACCEPT 5 packets, 756 bytes)
num   pkts bytes target     prot opt in     out     source               destinat
root@debian:~#
```

Last, but not least, you will need to remember that in order for the firewall rules to be persistent, you will need to save them to a file and then restore them automatically upon boot (using the preferred method of your choice or the one that is available for your distribution).

Saving firewall rules:

```
Ubuntu: iptables-save > /etc/iptables/rules.v4 # Ubuntu
iptables-save > /etc/sysconfig/iptables # CentOS / openSUSE
```

Restoring rules:

```
Ubuntu: iptables-restore < /etc/iptables/rules.v4 # Ubuntu
iptables-restore < /etc/sysconfig/iptables # CentOS / openSUSE
```

Here we can see a similar procedure (saving and restoring firewall rules by hand) using a dummy file called iptables.dump instead of the default one as shown above.

```
root@debian:~# iptables-save > iptables.dump
root@debian:~# iptables -F
root@debian:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@debian:~# iptables-restore < iptables.dump
root@debian:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
REJECT     tcp  --  anywhere             anywhere             tcp dpt:http reject-with icmp-port-unreachable
REJECT     tcp  --  anywhere             anywhere             tcp dpt:nfs reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@debian:~#
```

To make these changes persistent across boots:

Ubuntu: Install the **iptables-persistent** package, which will load the rules saved in the /etc/iptables/rules.v4 file.

CentOS: Add the following 2 lines to /etc/sysconfig/iptables-config:

```
IPTABLES_SAVE_ON_STOP="yes"
IPTABLES_SAVE_ON_RESTART="yes"
```

openSUSE: List allowed ports, protocols, addresses, and so forth (separated by commas) in /etc/sysconfig/SuSEfirewall2. For more information refer to the file itself, which is heavily commented.

Summary

The examples provided in this article, while not covering all the bells and whistles of iptables, serve the purpose of illustrating how to enable and disable traffic incoming or outgoing traffic.

Chapter 28: Static and dynamic routing

As we have anticipated in previous tutorials of this series, in this article we will discuss the routing of IP traffic statically and dynamically with specific applications.

First things first, let's get some definitions straight:

- In simple words, a packet is the basic unit that is used to transmit information within a network. Networks that use TCP/IP as network protocol follow the same rules for transmission of data: the actual information is split into packets that are made of both data and the address where it should be sent to.
- Routing is the process of "guiding" the data from source to destination inside a network.
- Static routing requires a manually-configured set of rules defined in a routing table. These rules are fixed and are used to define the way a packet must go through as it travels from one machine to another.
- Dynamic routing, or smart routing (if you wish), means that the system can alter automatically, as needed, the route that a packet needs to follow. However, in the context of the LFCE exam, the term dynamic routing refers to the ability to performing routing "on-the-fly" with the **ip** command.

IP and network device configuration

The iproute package provides a set of tools to manage networking and traffic control. We will use it throughout this article as they represent the replacement of legacy tools such as ifconfig and route.

The central utility in the **iproute** suite is called simply **ip**. Its basic syntax is as follows:

```
ip object command
```

where object can be only one of the following (only the most frequent objects are shown - you can refer to man ip for a complete list):

- **link**: network device.
- **addr**: protocol (IP or IPv6) address on a device.
- **route**: routing table entry.
- **rule**: rule in routing policy database.

whereas command represents a specific action that can be performed on object. You can run the following command to display the complete list of commands that can be applied to a particular object:

```
ip object help
```

For example,

```
ip link help
```

```
root@dev2:~# ip link help
Usage: ip link add [link DEV] [ name ] NAME
                   [ txqueuelen PACKETS ]
                   [ address LLADDR ]
                   [ broadcast LLADDR ]
                   [ mtu MTU ]
                   type TYPE [ ARGS ]
       ip link delete DEV type TYPE [ ARGS ]

       ip link set { dev DEVICE | group DEVGROUP } [ { up | down } ]
                         [ arp { on | off } ]
                         [ dynamic { on | off } ]
                         [ multicast { on | off } ]
                         [ allmulticast { on | off } ]
                         [ promisc { on | off } ]
                         [ trailers { on | off } ]
                         [ txqueuelen PACKETS ]
                         [ name NEWNAME ]
```

The above image shows, for example, that you can change the status of a network interface with the following command:

```
ip link set interface {up | down}
```

Example 1: Disabling and enabling a network interface

In this example we will disable and enable eth1:

```
ip link show
ip link set eth1 down
```

```
root@dev2:~# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN mode DEFAULT

    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mo
de DEFAULT qlen 1000
    link/ether 08:00:27:92:78:6e brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mo
de DEFAULT qlen 1000
    link/ether 08:00:27:fd:a6:b8 brd ff:ff:ff:ff:ff:ff
root@dev2:~# ip link set eth1 down
root@dev2:~# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN mode DEFAULT

    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mo
de DEFAULT qlen 1000
    link/ether 08:00:27:92:78:6e brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT
 qlen 1000
    link/ether 08:00:27:fd:a6:b8 brd ff:ff:ff:ff:ff:ff
root@dev2:~#
```

If you want to re-enable eth1,

```
ip link set eth1 up
```

Instead of displaying all the network interfaces, we can specify one of them:

```
ip link show eth1
```

which will return all the information for eth1.

Example 2: Displaying the main routing table

You can view your current main routing table with either of the following 3 commands:

```
ip route show
route -n
netstat -rn
```

```
root@dev2:~# ip route show
10.0.0.0/24 dev eth1  proto kernel  scope link  src 10.0.0.15
192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.0.15
root@dev2:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
root@dev2:~# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.0.0.0        0.0.0.0         255.255.255.0   U         0 0          0 eth1
192.168.0.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
root@dev2:~#
```

The first column in the output of the three commands indicates the target network. The output of ip route show (following the keyword dev) also presents the network devices that serve as physical gateway to those networks. Although nowadays the ip command is preferred over route, you can still refer to man ip-route and man route for a detailed explanation of the rest of the columns.

Example 3: Using a Linux server to route packages between two private networks

We want to route icmp (ping) packets from dev2 to dev4 and the other way around as well (note that both client machines are on different networks). The name of each NIC, along with its corresponding IPv4 address, is given inside square brackets.

Our test environment is as follows:

- Client 1: CentOS 7 [enp0s3: 192.168.0.17/24] - dev1
- Router: Debian Wheezy 7.7 [eth0: 192.168.0.15/24, eth1: 10.0.0.15/24] - dev2
- Client 2: openSUSE 13.2 [enp0s3: 10.0.0.18/24] - dev4

Let's view the routing table in dev1 (CentOS box):

```
ip route show
```

and then modify it in order to use its enp0s3 NIC and the connection to 192.168.0.15 to access hosts in the 10.0.0.0/24 network:

```
ip route add 10.0.0.0/24 via 192.168.0.15 dev enp0s3
```

Which essentially reads, "Add a route to the 10.0.0.0/24 network through the enp0s3 network interface using 192.168.0.15 as gateway".

```
[root@dev1 ~]# ip address show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
qlen 1000
    link/ether 08:00:27:f3:d4:74 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.17/24 brd 192.168.0.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fef3:d474/64 scope link
       valid_lft forever preferred_lft forever
[root@dev1 ~]# ip route show
192.168.0.0/24 dev enp0s3  proto kernel  scope link  src 192.168.0.17
[root@dev1 ~]#
```

```
[root@dev1 ~]# ip route add 10.0.0.0/24 via 192.168.0.15 dev enp0s3
[root@dev1 ~]# ip route show
10.0.0.0/24 via 192.168.0.15 dev enp0s3
192.168.0.0/24 dev enp0s3  proto kernel  scope link  src 192.168.0.17
[root@dev1 ~]#
```

Likewise in dev4 (openSUSE box) to ping hosts in the 192.168.0.0/24 network

```
ip route add 192.168.0.0/24 via 10.0.0.15 dev enp0s3
```

```
dev4:~ # ip address show enp0s3
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:1c:a1:75 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.18/24 brd 10.0.0.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe1c:a175/64 scope link
       valid_lft forever preferred_lft forever
dev4:~ # ip route show
10.0.0.0/24 dev enp0s3  proto kernel  scope link  src 10.0.0.18
dev4:~ # ip route add 192.168.0.0/24 via 10.0.0.15 dev enp0s3
dev4:~ # ip route show
10.0.0.0/24 dev enp0s3  proto kernel  scope link  src 10.0.0.18
192.168.0.0/24 via 10.0.0.15 dev enp0s3
dev4:~ #
```

Finally, we need to enable forwarding in our Debian router:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Now let's ping:

```
dev4:~ # ping -c 2 192.168.0.17
PING 192.168.0.17 (192.168.0.17) 56(84) bytes of data.
64 bytes from 192.168.0.17: icmp_seq=1 ttl=63 time=1.90 ms
64 bytes from 192.168.0.17: icmp_seq=2 ttl=63 time=1.45 ms

--- 192.168.0.17 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 1.451/1.676/1.901/0.225 ms
dev4:~ #
```

and

```
[root@dev1 ~]# ping -c 2 10.0.0.18
PING 10.0.0.18 (10.0.0.18) 56(84) bytes of data.
64 bytes from 10.0.0.18: icmp_seq=1 ttl=63 time=1.31 ms
64 bytes from 10.0.0.18: icmp_seq=2 ttl=63 time=1.28 ms

--- 10.0.0.18 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.287/1.299/1.311/0.012 ms
[root@dev1 ~]# _
```

To make these settings persistent across boots, edit /etc/sysctl.conf on the router and make sure the net.ipv4.ip_forward variable is set to true as follows:

```
net.ipv4.ip_forward = 1
```

In addition, configure the NICs on both clients (look for the configuration file within /etc/sysconfig/network on openSUSE and /etc/sysconfig/network-scripts on CentOS - in both cases it's called ifcfg-enp0s3).

Here's the configuration file from the openSUSE box:

```
BOOTPROTO=static
BROADCAST=10.0.0.255
IPADDR=10.0.0.18
NETMASK=255.255.255.0
GATEWAY=10.0.0.15
NAME=enp0s3
NETWORK=10.0.0.0
ONBOOT=yes
```

Example 4: Using a Linux server to route packages between a private network and the Internet

Another scenario where a Linux machine can be used as router is when you need to share your Internet connection with a private LAN.

- Router: Debian Wheezy 7.7 [eth0: Public IP, eth1: 10.0.0.15/24] - dev2
- Client: openSUSE 13.2 [enp0s3: 10.0.0.18/24] - dev4

In addition to set up packet forwarding and the static routing table in the client as in the previous example, we need to add a few iptables rules in the router:

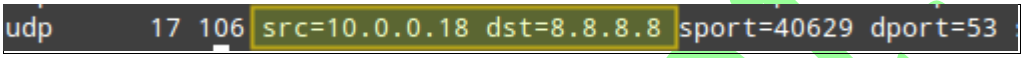
```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j
ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

The first command adds a rule to the POSTROUTING chain in the nat (Network Address Translation) table, indicating that the eth0 NIC should be used as the "exit door" for outgoing packages. MASQUERADE indicates that this NIC has a dynamic IP and that before sending the package to the "wild wild world" of the Internet, the private source address of the packet has to be changed to that of the public IP of the router. In a LAN with many hosts, the router keeps track of established connections in /proc/net/ip_conntrack so it knows where to return the response from the Internet to.

Only part of the output of

```
cat /proc/net/ip_conntrack
```

is show in the following screenshot.



where the origin (private IP of openSUSE box) and destination (Google DNS) of packets is highlighted. This was the result of running

```
curl www.tecmint.com
```

on the openSUSE box.

As I'm sure you can already guess, the router is using Google's 8.8.8.8 as nameserver, which explains why the destination of outgoing packets points to that address.

Note that incoming packages from the Internet are only accepted is if they are part of an already established connection (command #2), while outgoing packages are allowed "free exit" (command #3).

Don't forget to make your iptables rules persistent following the steps outlined Chapter 27 ("The firewall").

Summary

In this article we have explained how to set up static and dynamic routing, using a Linux box router(s). Feel free to add as many routers as you wish, and to experiment as much as you want. Do not hesitate to get back to us using the contact form below if you have any comments or questions.

Chapter 29: Encrypted filesystems and swap space

The idea behind encryption is to allow only trusted persons to access your sensitive data and to protect it from falling into the wrong hands in case of loss or theft of your machine / hard disk.

In simple terms, a key is used to "lock" access to your information so that it becomes available when the system is running and unlocked by an authorized user. This implies that if a person tries to examine the disk contents (plugging it to his own system or by booting the machine with a LiveCD/DVD/USB), he will only find unreadable data instead of the actual files.

In this article we will discuss how to set up encrypted file systems with dm-crypt (short for device mapper and cryptographic), the standard kernel-level encryption tool. Please note that since dm-crypt is a block-level tool, it can only be used to encrypt full devices, partitions, or loop devices (will not work on regular files or directories).

Preparing a drive / partition / loop device for encryption

Since we will wipe all data present in our chosen drive (/dev/sdb), first off we need to perform a backup of any important files contained in that partition BEFORE proceeding further.

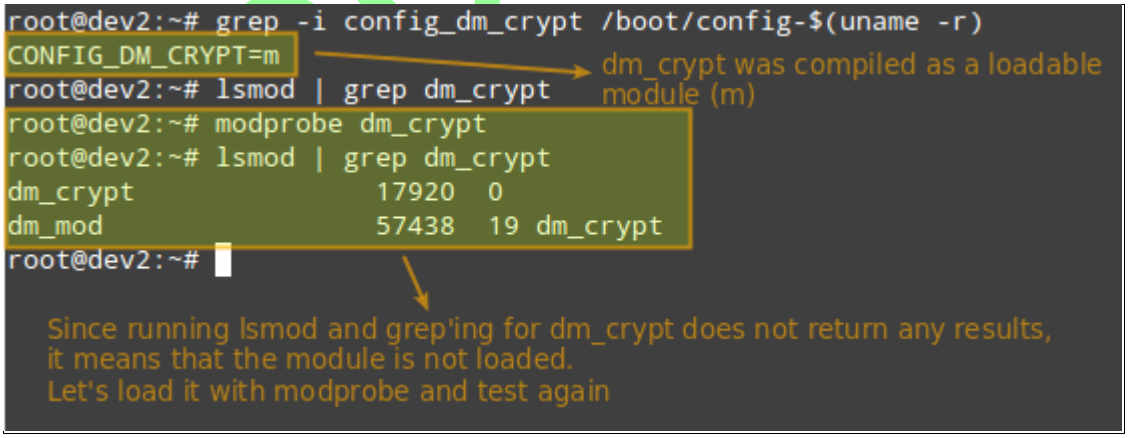
Wipe all data from /dev/sdb. We are going to use dd here, but you could also do it with other tools such as shred. Next, we will create a partition on this device, /dev/sdb1, following the explanation in Part 4 of the LFCS series.

```
dd if=/dev/urandom of=/dev/sdb bs=4096
```

Testing for encryption support

Before we proceed further, we need to make sure that our kernel has been compiled with encryption support:

```
grep -i config_dm_crypt /boot/config-$(uname -r)
```



As outlined in the image above, the dm-crypt kernel module needs to be loaded in order to set up encryption.

Installing cryptsetup

Cryptsetup is a frontend interface for creating, configuring, accessing, and managing encrypted file systems using dm-crypt.

```
aptitude update && aptitude install cryptsetup # Ubuntu
yum update && yum install cryptsetup # CentOS
zypper refresh && zypper install cryptsetup # openSUSE
```

Setting up an encrypted partition

The default operating mode for cryptsetup is LUKS (Linux Unified Key Setup) so we'll stick with it. We will begin by setting the LUKS partition and the passphrase:

```
cryptsetup -y luksFormat /dev/sdb1
```

```
root@dev2:~# cryptsetup -y luksFormat /dev/sdb1

WARNING!
========
This will overwrite data on /dev/sdb1 irrevocably.

Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase:
Verify passphrase:
root@dev2:~#
```

The command above runs cryptsetup with default parameters, which can be listed with

```
cryptsetup --version
```

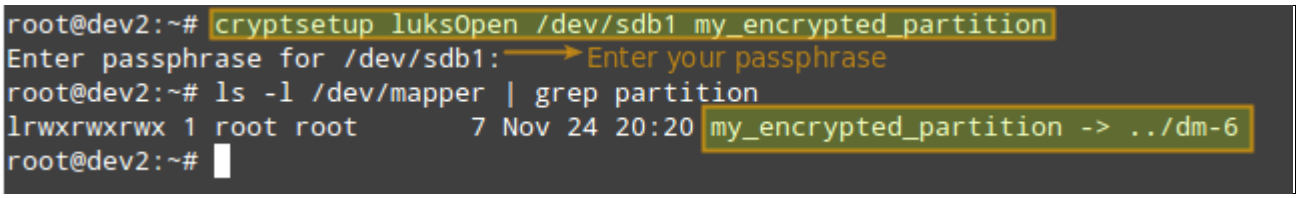
```
Default compiled-in keyfile parameters:
        Maximum keyfile size: 8192kB, Maximum interactive passphrase length 512 (characters)

Default compiled-in device cipher parameters:
        loop-AES: aes, Key 256 bits
        plain: aes-cbc-essiv:sha256, Key: 256 bits, Password hashing: ripemd160
        LUKS1: aes-cbc-essiv:sha256, Key: 256 bits, LUKS header hashing: sha1, RNG: /dev/urandom
root@dev2:~#
```

Should you want to change the cipher, hash, or key parameters, you can use the --cipher, --hash, and --key-size flags, respectively, with the values taken from /proc/crypto.

Next, we need to open the LUKS partition (we will be prompted for the passphrase that we entered earlier). If the authentication succeeds, our encrypted partition will be available inside /dev/mapper with the specified name:

```
cryptsetup luksOpen /dev/sdb1 my_encrypted_partition
```

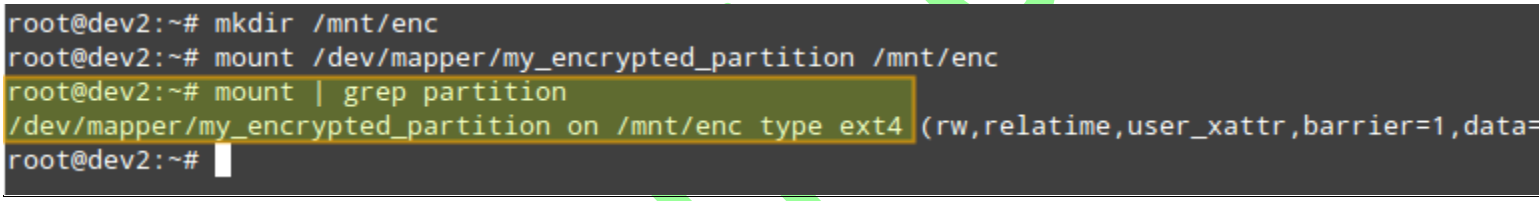


Now, we'll format out partition as ext4:

```
mkfs.ext4 /dev/mapper/my_encrypted_partition
```

and create a mount point to mount the encrypted partition. Finally, we may want to confirm whether the mount operation succeeded:

```
mkdir /mnt/enc
mount /dev/mapper/my_encrypted_partition /mnt/enc
mount | grep partition
```



When you are doing writing to or reading from your encrypted file system, simply unmount it

```
umount /mnt/enc
```

and close the LUKS partition

```
cryptesetup luksClose my_encrypted_partition
```

Testing encryption

Finally, we will check whether our encrypted partition is safe:

1. Open the LUKS partition

```
cryptsetup luksOpen /dev/sdb1 my_encrypted_partition
```

2. Enter your passphrase
3. Mount the partition

```
mount /dev/mapper/my_encrypted_partition /mnt/enc
```

4. Create a dummy file inside the mount point

```
echo "This is Part 3 of a 12-article series about the LFCE certification" >
/mnt/enc/testfile.txt
```

5. Verify that you can access the file that you just created

```
cat /mnt/enc/testfile.txt
```

6. Unmount the file system

```
umount /mnt/enc
```

7. Close the LUKS partition

```
cryptsetup luksClose my_encrypted_partition
```

8. Try to mount the partition as a regular file system. It should indicate an error.

```
mount /dev/sdb1 /mnt/enc
```

```
root@dev2:~# cryptsetup luksOpen /dev/sdb1 my_encrypted_partition 1
Enter passphrase for /dev/sdb1: 2
root@dev2:~# mount /dev/mapper/my_encrypted_partition /mnt/enc 3
root@dev2:~# echo "This is Part 3 of a 12-article series about the LFCE certification" > /mnt/enc/testfile.txt 4
root@dev2:~# cat /mnt/enc/testfile.txt 5
This is Part 3 of a 12-article series about the LFCE certification
root@dev2:~# umount /mnt/enc 6
root@dev2:~# cryptsetup luksClose my_encrypted_partition 7
root@dev2:~# mount /dev/sdb1 /mnt/enc 8
mount: unknown filesystem type 'crypto_LUKS'
root@dev2:~#
```

Encrypting the swap space for further security

The passphrase you entered earlier to use the encrypted partition is stored in RAM memory while it's open. If someone can get his hands on this key, he will be able to decrypt the data. This is especially easy to do in the case of a laptop, since while hibernating the contents of RAM are kept on the swap partition.

To avoid leaving a copy of your key accessible to a thief, encrypt the swap partition following these steps:

1. Create a partition to be used as swap with the appropriate size (/dev/sdd1 in our case) and encrypt it as explained earlier. Name it just "swap" for convenience.
2. Set it as swap and activate it

```
mkswap /dev/mapper/swap
swapon /dev/mapper/swap
```

3. Next, change the corresponding entry in /etc/fstab

```
/dev/mapper/swap none swap sw 0 0
```

4. Finally, edit /etc/crypttab and reboot

```
swap                /dev/sdd1          /dev/urandom swap
```

Once the system has finished booting, you can verify the status of the swap space:

```
cryptsetup status swap
```

```
root@dev2:~# cryptsetup status swap
/dev/mapper/swap is active and is in use.
  type:    PLAIN
  cipher:  aes-cbc-essiv:sha256
  keysize: 256 bits
  device:  /dev/mapper/debian-swap_1
  offset:  0 sectors
  size:    1146880 sectors
  mode:    read/write
root@dev2:~#
```

Summary

In this article we have explored how to encrypt a partition and swap space. With this setup, your data should be considerably safe.

Chapter 30: System usage, utilization, and troubleshooting

Although Linux is very reliable, wise system administrators should find a way to keep an eye on the system's behavior and utilization at all times. Ensuring an uptime as close to 100% as possible and the availability of resources are critical needs in many environments. Examining the past and current status of the system will allow us to foresee and most likely prevent possible issues.

In this article we will present a list of a few tools that are available in most upstream distributions to check on the system status, analyze outages, and troubleshoot ongoing issues. Specifically, of the myriad of available data, we will focus on CPU, storage space and memory utilization, basic process management, and log analysis.

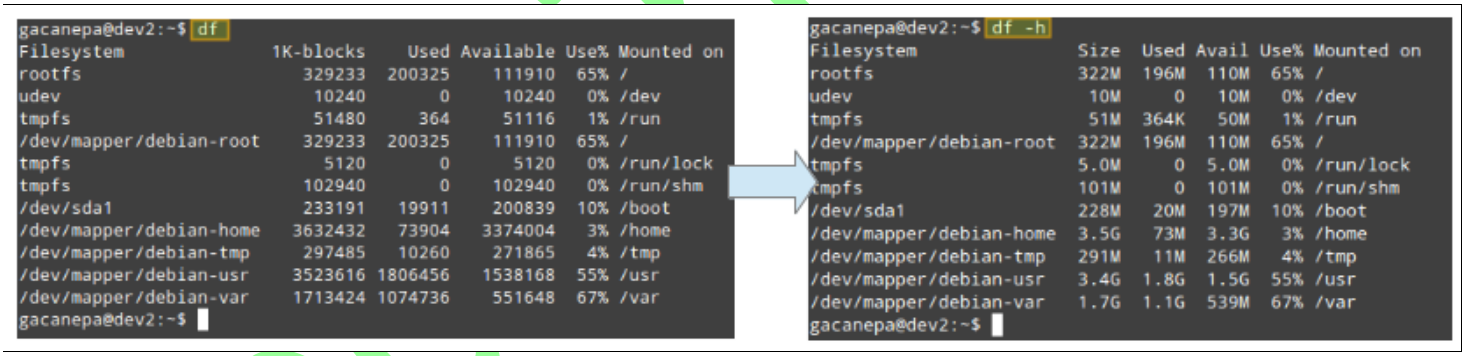
Storage space utilization

There are 2 well-known commands in Linux that are used to inspect storage space usage: **df** and **du**.

The first one, **df** (which stands for disk free), is typically used to report overall disk space usage by file system.

Example 1: Reporting disk space usage in bytes and human-readable format

Without options, df reports disk space usage in bytes. With the -h flag it will display the same information using MB or GB instead. Note that this report also includes the total size of each file system (in 1-K blocks), the free and available spaces, and the mount point of each storage device:



That's certainly nice - but there's another limitation that can render a file system unusable, and that is running out of inodes. All files in a file system are mapped to an inode that contains its metadata.

Example 2: Inspecting inode usage by file system in human-readable format

With

```
df -hTi
```

you can see the amount of used and available inodes:

```
gacanepa@dev2:~$ df -hTi
Filesystem              Type     Inodes IUsed IFree IUse% Mounted on
rootfs                  rootfs      84K  7.1K   76K    9% /
udev                    devtmpfs    62K   472   61K    1% /dev
tmpfs                   tmpfs       63K   492   63K    1% /run
/dev/mapper/debian-root ext4        84K  7.1K   76K    9% /
tmpfs                   tmpfs       63K     5   63K    1% /run/lock
tmpfs                   tmpfs       63K     2   63K    1% /run/shm
/dev/sda1               ext2       122K   241  122K    1% /boot
/dev/mapper/debian-home ext4       226K   146  226K    1% /home
/dev/mapper/debian-tmp  ext4        76K    18   76K    1% /tmp
/dev/mapper/debian-usr  ext4       219K   69K  150K   32% /usr
/dev/mapper/debian-var  ext4       107K   31K   76K   29% /var
gacanepa@dev2:~$
```

According to the above image, there are 146 used inodes (1%) in /home, which means that you can still create 226K files in that file system.

Example 3: Finding and / or deleting empty files and directories

Note that you can run out of storage space long before running out of inodes, and viceversa. For that reason, you need to monitor not only the storage space utilization but also the number of inodes used by file system.

Use the following commands to find empty files or directories (which occupy 0 B) that are using inodes without a reason:

```
find  /home -type f -empty
find  /home -type d -empty
```

Also, you can add the -delete flag at the end of each command if you also want to delete those empty files and directories:

```
root@dev2:~# find  /home -type f -empty
/home/gacanepa/Inxs.mp4
/home/gacanepa/.aptitude/config
/home/gacanepa/mail/.imap/dovecot-uidvalidity.54987a50
/home/jdoe/mail/.imap/dovecot-uidvalidity.5498d61c
root@dev2:~# find  /home -type f -empty -delete
root@dev2:~# find  /home -type f -empty
root@dev2:~#
```

The previous procedure deleted 4 files. Let's check again the number of used / available nodes again in /home:

```
gacanepa@dev2:~$ df -hTi | grep home
/dev/mapper/debian-home ext4       226K   142  226K    1% /home
gacanepa@dev2:~$
```

As you can see, there are 142 used inodes now (4 less than before).

Example 4: Examining disk usage by directory

If the use of a certain file system is above a predefined percentage, you can use du (short for disk usage) to find out what are the files that are occupying the most space. The example is given for /var, which as you can see in the first image above, is used at its 67%.

```
du -sch /var/*
```

```
root@dev2:~# du -sch /var/*
8.4M    /var/archives
3.9M    /var/backups
552M    /var/cache
248M    /var/lib
4.0K    /var/local
0       /var/lock
43M     /var/log
16K     /var/lost+found
40K     /var/mail
4.0K    /var/opt
0       /var/run
28M     /var/spool
4.0K    /var/tmp
135M    /var/www
1016M   total
root@dev2:~#
```

Note that you can switch to any of the above subdirectories to find out exactly what's in them and how much each item occupies. You can then use that information to either delete some files if there are not needed or extend the size of the logical volume if necessary.

Memory and CPU utilization

The classic tool in Linux that is used to perform an overall check of CPU / memory utilization and process management is top. In addition, top displays a real-time view of a running system. There other tools that could be used for the same purpose, such **htop**, but I have settled for top because it is installed out-of-the-box in any Linux distribution.

Example 5: Displaying a live status of your system with top

To start top, simply type

```
top
```

in your command line, and hit Enter.

Let's examine a typical top output:

```
top - 20:41:32 up  7:41,  1 user,  load average: 0.00, 0.01, 0.05 1
Tasks: 121 total,   1 running, 120 sleeping,   0 stopped,   0 zombie 2
%Cpu(s):  0.0 us,  0.7 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st 3
KiB Mem:    514792 total,   432576 used,    82216 free,   115544 buffers 4
KiB Swap:   573436 total,        0 used,   573436 free,   123540 cached 5

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 5096 root      20   0  4424 1420 1036 R   1.0  0.3   0:00.06 top
 2804 mysql     20   0  317m  39m 5880 S   0.3  7.8   0:22.42 mysqld
    1 root      20   0  2196  752  648 S   0.0  0.1   0:01.17 init          6
    2 root      20   0     0    0    0 S   0.0  0.0   0:00.01 kthreadd
    3 root      20   0     0    0    0 S   0.0  0.0   0:00.64 ksoftirqd/0
    6 root      rt   0     0    0    0 S   0.0  0.0   0:00.26 watchdog/0
    7 root      0  20     0    0    0 S   0.0  0.0   0:00.00 cpuset
```

In rows 1 through 5 the following information is displayed:

1. The current time (8:41:32 pm) and uptime (7 hours and 41 minutes). Only one user is logged on to the system, and the load average during the last 1, 5, and 15 minutes, respectively. 0.00, 0.01, and 0.05 indicate that over those time intervals, the system was idle for 0% of the time (0.00: no processes were waiting for the CPU), it then was overloaded by 1% (0.01: an average of 0.01 processes were waiting for the CPU) and 5% (0.05). If less than 0 and the smaller the number (0.65, for example), the system was idle for 35% during the last 1, 5, or 15 minutes, depending where 0.65 appears.
2. Currently there are 121 processes running (you can see the complete listing in 6). Only 1 of them is running (top in this case, as you can see in the %CPU column) and the remaining 120 are waiting in the background but are "sleeping" and will remain in that state until we call them. How? You can verify this by opening a mysql prompt and execute a couple of queries. You will notice how the number of running processes increases. Alternatively, you can open a web browser and navigate to any given page that is being served by apache and you will get the same result. Of course, these examples assume that both services are installed in your server.
3. us (time running user processes with unmodified priority), sy (time running kernel processes), ni (time running user processes with modified priority), wa (time waiting for I/O completion), hi (time spent servicing hardware interrupts), si (time spent servicing software interrupts), st (time stolen from the current vm by the hypervisor - only in virtualized environments).
4. Physical memory usage
5. Swap space usage

Example 6: Inspecting physical memory usage

To inspect RAM memory and swap usage you can also use **free**:

```
root@dev2:~# free
             total       used       free     shared    buffers     cached
Mem:        514792     478844      35948          0     116724     124332
-/+ buffers/cache:     237788     277004
Swap:       573436         44     573392
root@dev2:~#
```

Of course you can also use the -m (MB) or -g (GB) switches to display the same information in human-readable form:

```
root@dev2:~# free -m
             total       used       free     shared    buffers     cached
Mem:           502        467         35          0        114        121
-/+ buffers/cache:        232        270
Swap:          559          0        559
root@dev2:~#
```

Either way, you need to be aware of the fact that the kernel reserves as much memory as possible and makes it available to processes when they request it. Particularly, the "-/+ buffers/cache" line shows the actual values after this I/O cache is taken into account. In other words, the amount of memory used by processes and the amount available to other processes (in this case, 232 MB used and 270 MB available, respectively). When processes need this memory, the kernel will automatically decrease the size of the I/O cache.

A closer look at processes

At any given time, there many processes running on our Linux system. There are two tools that we will use to monitor processes closely: ps and pstree.

Example 7: Displaying the whole process list in your system with ps (full standard format)

Using the -e and -f options combined into one (-ef) you can list all the processes that are currently running on your system. You can pipe this output to other tools, such as **grep** to narrow down the output to your desired process(es):

```
gacanepa@dev2:~$ ps -ef | grep -i squid | grep -v grep
root      3157     1  0 Jan02 ?        00:00:00 /usr/sbin/squid -D -YC
proxy     3162  3157  0 Jan02 ?        00:00:05 (squid) -D -YC
proxy     3181  3162  0 Jan02 ?        00:00:00 (squidGuard)
proxy     3182  3162  0 Jan02 ?        00:00:00 (squidGuard)
proxy     3183  3162  0 Jan02 ?        00:00:00 (squidGuard)
proxy     3184  3162  0 Jan02 ?        00:00:00 (squidGuard)
proxy     3185  3162  0 Jan02 ?        00:00:00 (squidGuard)
gacanepa@dev2:~$
```

The process listing above shows the following information: owner of the process, PID, Parent PID (the parent process), processor utilization, time when command started, tty (the ? indicates it's a daemon), the cumulated CPU time, and the command associated with the process.

Example 8: Customizing and sorting the output of **ps**

However, perhaps you don't need all that information, and would like to show the owner of the process, the command that started it, its PID and PPID, and the percentage of memory it's currently using - in that order, and sort by memory use in descending order (note that ps by default is sorted by PID).

```
ps -eo user,comm,pid,ppid,%mem --sort -%mem
```

where the minus sign in front of %mem indicates sorting in descending order.

```
gacanepa@dev2:~$ ps -eo user,comm,pid,ppid,%mem --sort -%mem
USER     COMMAND            PID  PPID %MEM
mysql    mysqld            2806  2479  7.8
root     apache2           2190     1  2.3
colord   colord-sane       2394     1  1.7
backuppc BackupPC          2292     1  1.7
www-data apache2           2289  2190  1.0
www-data apache2           2286  2190  1.0
www-data apache2           2287  2190  1.0
www-data apache2           2288  2190  1.0
www-data apache2           2290  2190  1.0
backuppc BackupPC_trashC   2317  2292  0.9
proxy    squid             3162  3157  0.9
colord   colord            2390     1  0.7
root     console-kit-dae   3578     1  0.7
root     sshd              3575  3194  0.6
root     cupsd             2388     1  0.6
proxy    squidGuard        3181  3162  0.5
proxy    squidGuard        3182  3162  0.5
```

If for some reason a process starts taking too much system resources and it's likely to jeopardize the overall functionality of the system, you will want to stop or pause its execution passing one of the following signals using the kill program to it. Other reasons why you would consider doing this is when you have started a process in the foreground but want to pause it and resume in the background.

Signal name	Signal number	Description
SIGTERM	15	Kill the process gracefully.
SIGINT	2	This is the signal that is sent when we press Ctrl + C. It aims to interrupt the process, but the process may ignore it.
SIGKILL	9	This signal also interrupts the process but it does so unconditionally (use with care!) since a process cannot ignore it.
SIGHUP	1	Short for "Hang UP", this signals instructs daemons to reread its configuration file without actually stopping the process.
SIGTSTP	20	Pause execution and wait ready to continue. This is the signal that is sent when we type the Ctrl + Z key combination.
SIGSTOP	19	The process is paused and doesn't get any more attention from the CPU cycles until it is restarted.
SIGCONT	18	This signal tells the process to resume execution after having received either SIGTSTP or SIGSTOP. This is the signal that is sent by the shell when we use the fg or bg commands.

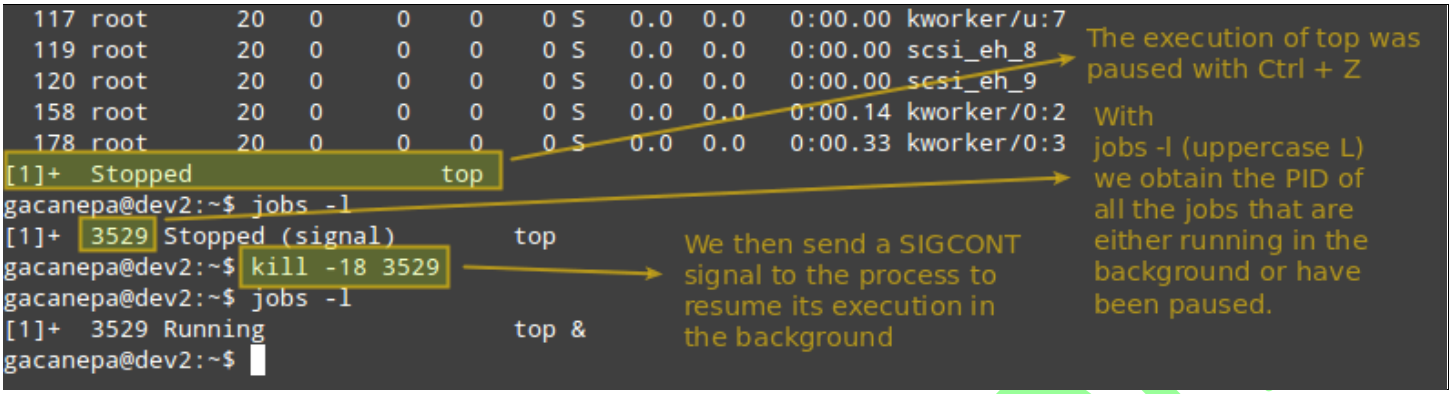
Example 9: Pausing the execution of a running process and resuming it in the background

When the normal execution of a certain process implies that no output will be sent to the screen while it's running, you may want to either start it in the background (appending an ampersand at the end of the command)

```
process_name &
```

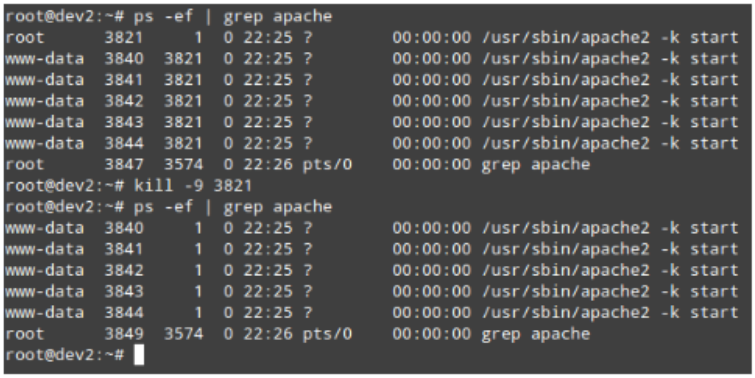
or, once it has started running in the foreground, pause it and send it to the background with

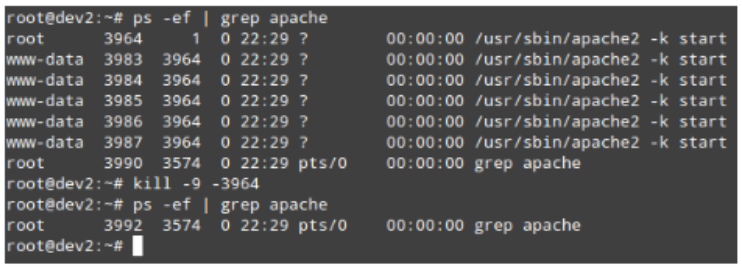
```
Ctrl + Z
kill -18 PID
```



Example 10: Killing by force a process "gone wild"

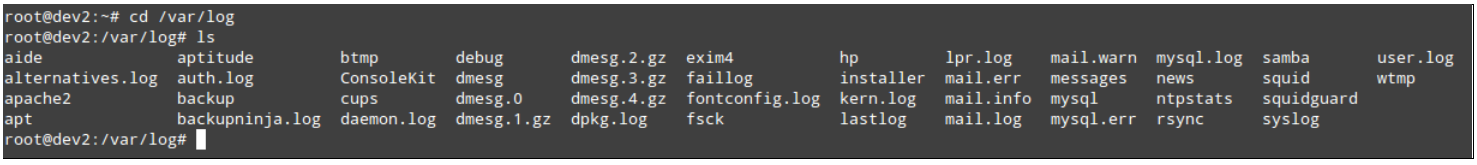
Please note that each distribution provides tools to gracefully stop / start / restart / reload common services, such as service in SysV-based systems or systemctl in systemd-based systems. If a process does not respond to those utilities, you can kill it by force by sending it the SIGKILL signal to it.





So... what happened / is happening?

When there has been any kind of outage in the system (be it a power outage, a hardware failure, a planned or unplanned interruption of a process, or any abnormality at all), the logs in /var/log are your best friends to determine what happened or what could be causing the issues you're facing.



Some of the items in /var/log are regular text files, others are directories, and yet others are compressed files of rotated (historical) logs. You will want to check those with the word error in their name, but inspecting the rest can come in handy as well.

Example 11: Examining logs for errors in processes

Picture this scenario. Your LAN clients are unable to print to network printers. The first step to troubleshoot this situation is going to /var/log/cups directory and see what's in there. You can use the tail command to display the last 10 lines of the error_log file, or tail -f error_log for a real-time view of the log.

```
root@dev2:~# cd /var/log/cups
root@dev2:/var/log/cups# ls
access_log  cups-pdf_log  error_log  page_log
root@dev2:/var/log/cups# tail error_log
W [30/Dec/2014:23:38:07 -0300] Please move "SystemGroup lpadmin" on line 16 of /etc/cups/cupsd.conf to the /etc/
a future release.
E [31/Dec/2014:04:34:29 -0300] Avahi client failed, closing client to allow a clean restart
W [31/Dec/2014:07:36:08 -0300] Please move "SystemGroup lpadmin" on line 16 of /etc/cups/cupsd.conf to the /etc/
a future release.
E [31/Dec/2014:10:14:04 -0300] Avahi client failed, closing client to allow a clean restart
W [01/Jan/2015:13:00:45 -0300] Please move "SystemGroup lpadmin" on line 16 of /etc/cups/cupsd.conf to the /etc/
a future release.
E [01/Jan/2015:22:51:45 -0300] Avahi client failed, closing client to allow a clean restart
W [02/Jan/2015:21:42:49 -0300] Please move "SystemGroup lpadmin" on line 16 of /etc/cups/cupsd.conf to the /etc/
a future release.
E [03/Jan/2015:01:52:01 -0300] Avahi client failed, closing client to allow a clean restart
W [03/Jan/2015:21:59:50 -0300] Please move "SystemGroup lpadmin" on line 16 of /etc/cups/cupsd.conf to the /etc/
a future release.
E [03/Jan/2015:21:59:50 -0300] Unable to bind broadcast socket - Address already in use.
root@dev2:/var/log/cups#
```

The above screenshot provides some helpful information to understand what could be causing your issue. Note that following the steps or correcting the malfunctioning of the process still may not solve the overall problem, but if you become used right from the start to check on the logs every time a problem arises (be it a local or a network one) you'll be definitely on the right track.

Example 12: Examining the logs for hardware failures

Although hardware failures can be tricky to troubleshoot, you should check the dmesg and messages logs and grep for related words to a hardware part presumed faulty.

The image below is taken from /var/log/messages after looking for the word error using the following command:

```
less /var/log/messages | grep -i error
```

We can see that we're having a problem with two storage devices: /dev/sdb and /dev/sdc, which in turn cause an issue with the RAID array.

```
Sep 11 22:42:20 debian kernel: [ 3530.862653] scsi 4:0:0:0: [sdb] Unhandled error code
Sep 11 22:42:20 debian kernel: [ 3530.886956] md: super_written gets error=-19, uptodate=0
Sep 11 22:42:20 debian kernel: [ 3530.894381] md: super_written gets error=-19, uptodate=0
Sep 11 22:53:31 debian kernel: [ 4201.040036] scsi 3:0:0:0: [sdc] Unhandled error code
Sep 11 23:05:28 debian kernel: [    5.233212] EXT4-fs (dm-0): re-mounted. Opts: errors=remount-ro
Sep 12 10:22:31 debian kernel: [    5.598627] EXT4-fs (dm-0): re-mounted. Opts: errors=remount-ro
Sep 13 22:53:35 debian kernel: [    6.157965] EXT4-fs (dm-0): re-mounted. Opts: errors=remount-ro
Sep 15 13:18:15 debian kernel: [    5.535049] EXT4-fs (dm-0): re-mounted. Opts: errors=remount-ro
Sep 18 21:11:25 debian kernel: [   10.332429] EXT4-fs (dm-0): re-mounted. Opts: errors=remount-ro
```

Summary

In this article we have explored some of the tools that can help you to always be aware of your system's overall status. In addition, you need to make sure that your operating system and installed packages are updated to their latest stable versions. And never, ever, forget to check the logs!

Chapter 31: Setting up a network repository

Installing, updating, and removing (when needed) installed programs are key responsibilities in a system administrator's daily life. When a machine is connected to the Internet, these tasks can be easily performed using a package management system such as aptitude (or apt-get), yum, or zypper, depending on your chosen distribution. You can also download standalone .deb or .rpm files and install them with dpkg or rpm, respectively.

However, when a machine does not have access to the world wide web, another methods are necessary. Why would anyone want to do that? The reasons range from saving Internet bandwidth (thus avoiding several concurrent connections to the outside) to securing packages compiled from source locally, and including the possibility of providing packages that for legal reasons (for example, software that is restricted in some countries) cannot be included in official repositories.

That is precisely where network repositories come into play, which is the central topic of this article.

Setting up a network repository server

As a first step, we will handle the installation and configuration of a CentOS 7 box as a repository server [IP address 192.168.0.17] and a CentOS 6.6 machine as client. The setup for openSUSE is almost identical. As for Ubuntu, there is a great article on this site that explains, step by step, how to set up your own, private repository.

Our first choice will be the way in which clients will access the repository server - FTP and HTTP are the most well used. This will also allow us to display the package listing using a web browser.

Next, we need to create directories to store the .rpm packages. We will create subdirectories within /var/www/html/repos accordingly. For our convenience, we may also want to create other subdirectories to host packages for different versions of each distribution (of course we can still add as many directories as needed later) and even different architectures.

An important thing to take into consideration when setting up your own repository is that you will need a considerable amount of available disk space (~20 GB). If you don't, resize the filesystem where you're planning on storing the repository's contents, or even better add an extra dedicated storage device to host the repository.

That being said, we will begin by creating the directories that we will need to host the repository:

```
mkdir -p /var/www/html/repos/centos/6/6
```

After we have created the directory structure for our repository server, we will initialize in /var/www/html/repos/centos/6/6 the database that keeps tracks of packages and their corresponding dependencies using createrepo.

Install createrepo if you haven't already done so:

```
yum update && yum install createrepo
```

Then initialize the database

```
createrepo /var/www/html/repos/centos/6/6
```

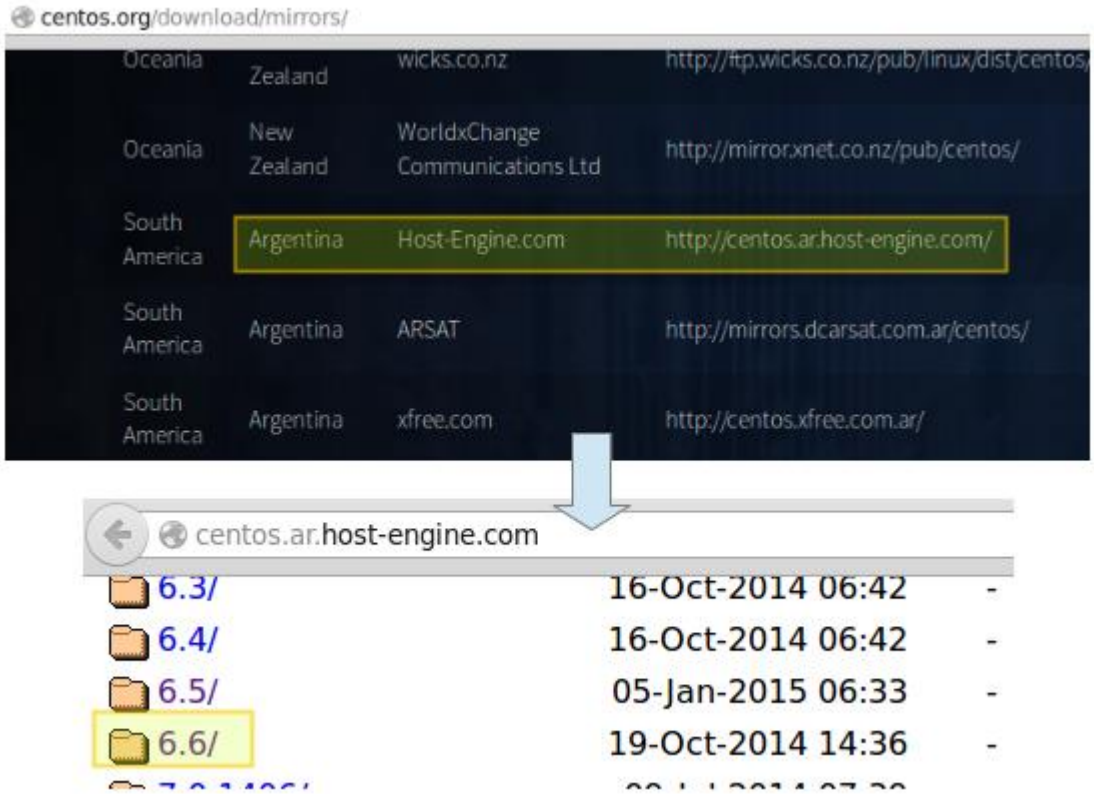
```
[root@dev1 ~]# createrepo /var/www/html/repos/centos/6/6
Saving Primary metadata
Saving file lists metadata
Saving other metadata
Generating sqlite DBs
Sqlite DBs complete
[root@dev1 ~]# ls -lR createrepo /var/www/html/repos/centos/6/6
ls: cannot access createrepo: No such file or directory
/var/www/html/repos/centos/6/6:
total 4
drwxr-xr-x. 2 root root 4096 Jan 20 17:52 repodata

/var/www/html/repos/centos/6/6/repodata:
total 28
-rw-r--r--. 1 root root  586 Jan 20 17:52 01a3b489a465bcac22a43492163df43451dc6ce47d27f66de289756b91635523-filelists.sqlite.bz2
-rw-r--r--. 1 root root  123 Jan 20 17:52 401dc19bda88c82c403423fb835844d64345f7e95f5b9835888189c03834cc93-filelists.xml.gz
-rw-r--r--. 1 root root 1131 Jan 20 17:52 5dc1e6e73c84803f059bb3065e684e56adfc289a7e398946574d79dac6643945-primary.sqlite.bz2
-rw-r--r--. 1 root root  123 Jan 20 17:52 6bf9672d0862e8ef8b8ff05a2fd0208a922b1f5978e6589d87944c88259cb670-other.xml.gz
-rw-r--r--. 1 root root  575 Jan 20 17:52 7c36572015e075add2b38b900837bcdbb8a504130ddff49b2351a7fc0affa3d4-other.sqlite.bz2
-rw-r--r--. 1 root root  134 Jan 20 17:52 dabe2ce5481d23de1f4f52bdcfee0f9af98316c9e0de2ce8123adeefa0dd08b9-primary.xml.gz
-rw-r--r--. 1 root root 2962 Jan 20 17:52 repomd.xml  ---> This file contains the metadata for our newly
[root@dev1 ~]#                                            created repository.
```

Updating the repository

Assuming that the repository server has access to the Internet, we will pull an online repository to get the latest updates of packages. If that is not the case, you can still copy the entire contents of the Packages directory from a CentOS 6.6 installation DVD. In this tutorial we will assume the first case.

In order to optimize our download speed, we will choose a CentOS 6.6 mirror from a location near us. Go to http://centos.org/download/mirrors/ and pick the one that is closer to your location (Argentina in my case):



Then, navigate to the os directory inside the highlighted link and then choose the appropriate architecture. Once there, copy the link in the address bar and download the contents to the dedicated directory in the repository server:


```
rsync -avz rsync://centos.ar.host-engine.com/6.6/os/x86_64/ /var/www/html/repos/centos/6/6/
```

In case that the chosen repository turns out to be offline for some reason, go back and choose a different one. No big deal.

Now is the time when you may want to relax and maybe watch an episode of your favorite TV show, because mirroring the online repository may take quite a while.

Once the download has completed, you can verify the usage of disk space with

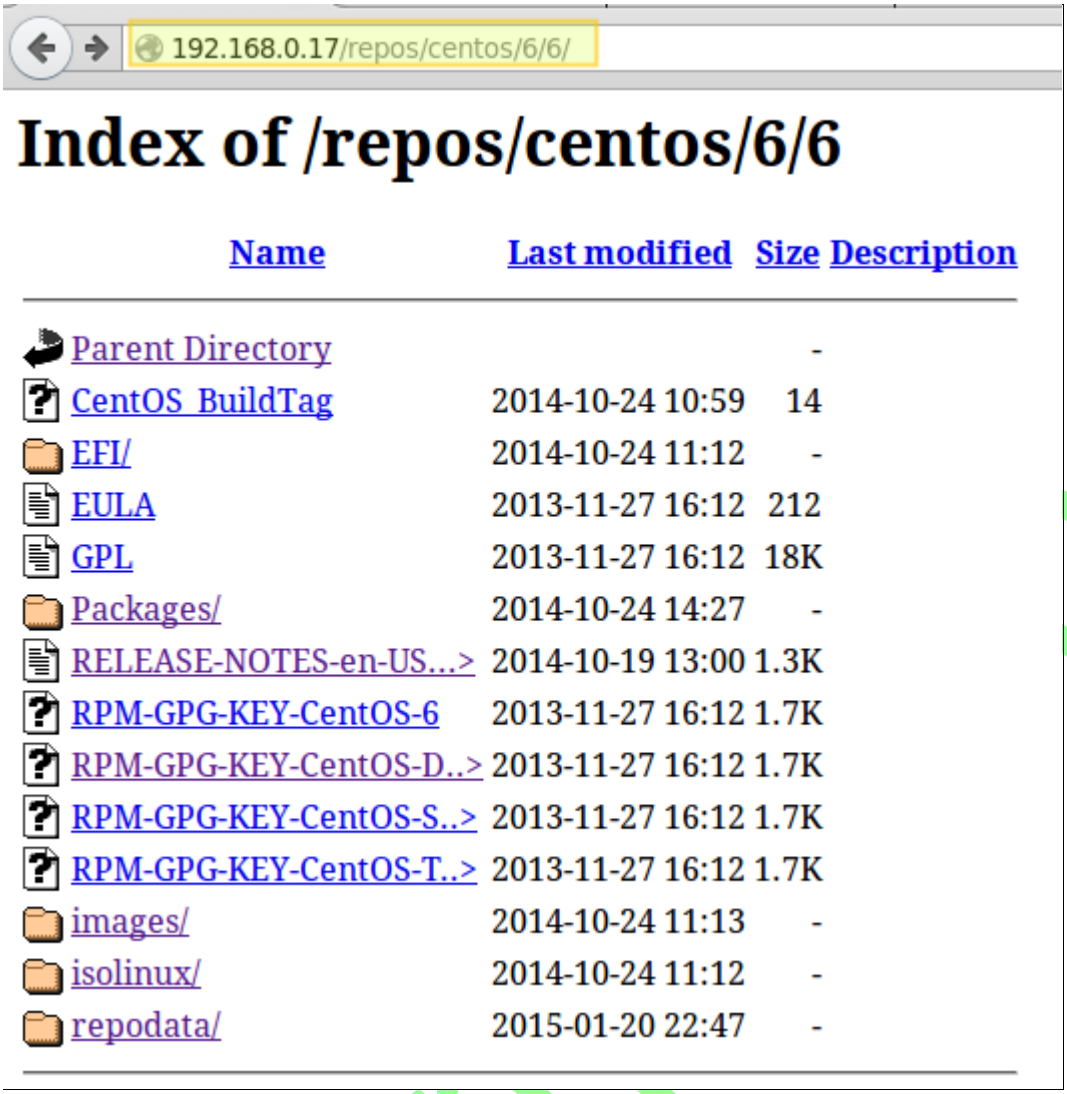
```
du -sch /var/www/html/repos/centos/6/6/*
```

```
[root@dev1 ~]# du -sch /var/www/html/repos/centos/6/6/*
4.0K    /var/www/html/repos/centos/6/6/CentOS_BuildTag
256K    /var/www/html/repos/centos/6/6/EFI
4.0K    /var/www/html/repos/centos/6/6/EULA
20K     /var/www/html/repos/centos/6/6/GPL
444M    /var/www/html/repos/centos/6/6/images
38M     /var/www/html/repos/centos/6/6/isolinux
5.3G    /var/www/html/repos/centos/6/6/Packages
4.0K    /var/www/html/repos/centos/6/6/RELEASE-NOTES-en-US.html
27M     /var/www/html/repos/centos/6/6/repodata
4.0K    /var/www/html/repos/centos/6/6/RPM-GPG-KEY-CentOS-6
4.0K    /var/www/html/repos/centos/6/6/RPM-GPG-KEY-CentOS-Debug-6
4.0K    /var/www/html/repos/centos/6/6/RPM-GPG-KEY-CentOS-Security-6
4.0K    /var/www/html/repos/centos/6/6/RPM-GPG-KEY-CentOS-Testing-6
5.8G    total
[root@dev1 ~]#
```

Finally, update the repository's database

```
createrepo --update /var/www/html/repos/centos/6/6
```

You may also want to launch your web browser and navigate to the repos/centos/6/6 directory so as to verify that you can see the contents:



And you're ready to go - now it's time to configure the client.

Configuring the client

You can add the configuration files for custom repositories in the /etc/yum.repos.d directory. Configuration files need to end in .repo and follow the same basic structure.

```
[repository_name]
Description
URL
```

Most likely, there will be already other .repo files in /etc/yum.repos.d. To properly test your repository, you can either delete those configuration files (not really recommended, since you may need them later) or rename them, as I did, by appending .orig to each file name:

```
cd /etc/yum.repos.d
for i in $(ls *.repo); do mv $i $i.orig; done
```

```
[root@dev3 ~]# cd /etc/yum.repos.d
[root@dev3 yum.repos.d]# ls
CentOS-Base.repo       CentOS-fasttrack.repo  CentOS-Vault.repo  epel-testing.repo  mirrors-rpmforge-extras   rpmforge.repo
CentOS-Debuginfo.repo  CentOS-Media.repo      epel.repo          mirrors-rpmforge   mirrors-rpmforge-testing
[root@dev3 yum.repos.d]# less CentOS-Base.repo
[root@dev3 yum.repos.d]# for i in $(ls *.repo); do mv $i $i.orig; done
[root@dev3 yum.repos.d]# ls
CentOS-Base.repo.orig       CentOS-fasttrack.repo.orig  CentOS-Vault.repo.orig  epel-testing.repo.orig  mirrors-rpmforge-extras   rpmforge.repo.orig
CentOS-Debuginfo.repo.orig  CentOS-Media.repo.orig      epel.repo.orig          mirrors-rpmforge        mirrors-rpmforge-testing
[root@dev3 yum.repos.d]#
```

In our case, we will name our configuration file as tecmint.repo and insert the following lines:

```
[tecmint]
name=Example repo for Part 11 of the LFCE series on Tecmint.com
baseurl=http://192.168.0.17/repos/centos/6/6/
gpgcheck=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-6
```

In addition, we have enabled GPG signature-checking for all packages in our repository.

Using the repository

Once the client has been properly configured, you can issue the usual yum commands to query the repository. Note how yum info subversion indicates that the information about the package is coming from our newly created repository:

```
[root@dev3 yum.repos.d]# yum info subversion
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Available Packages
Name        : subversion
Arch        : i686
Version     : 1.6.11
Release     : 10.el6_5
Size        : 2.2 M
Repo        : tecmint
Summary     : A Modern Concurrent Version Control System
URL         : http://subversion.apache.org/
License     : ASL 1.1
Description : Subversion is a concurrent version control system which enables one
            : or more users to collaborate in developing and maintaining a
            : hierarchy of files and directories while keeping a history of all
            : changes.  Subversion only stores the differences between versions,
            : instead of every complete file.  Subversion is intended to be a
            : compelling replacement for CVS.

Name        : subversion
Arch        : x86_64
Version     : 1.6.11
Release     : 10.el6_5
Size        : 2.3 M
Repo        : tecmint
Summary     : A Modern Concurrent Version Control System
URL         : http://subversion.apache.org/
License     : ASL 1.1
Description : Subversion is a concurrent version control system which enables one
            : or more users to collaborate in developing and maintaining a
            : hierarchy of files and directories while keeping a history of all
            : changes.  Subversion only stores the differences between versions,
            : instead of every complete file.  Subversion is intended to be a
            : compelling replacement for CVS.

[root@dev3 yum.repos.d]#
```

Or you can install or update an already installed package:

```
yum { install | update } package
```

For example,

```
yum update && yum install subversion
```

```
Dependencies Resolved

================================================================================
 Package                                    Arch
================================================================================
Installing:
 subversion                                 x86_64
Installing for dependencies:
 libproxy                                   x86_64
 libproxy-bin                               x86_64
 libproxy-python                            x86_64
 neon                                       x86_64
 pakchois                                   x86_64
 perl-URI                                   noarch

Transaction Summary
================================================================================
Install       7 Package(s)

Total download size: 2.6 M
Installed size: 12 M
Is this ok [y/N]: y
```

Keeping the repository up-to-date

To make sure our repository is always current, we need to synchronize it with the online repositories on a periodic basis. We can use rsync for this task as well (as explained in Part 6 of the LFCS series, rsync allows us to synchronize two directories, one local and one remote). Run the rsync command that was used to initially download the repository through a cron job and you're good to go. Remember to set the cron job to a time of the day when the update will not cause a negative impact in the available bandwidth.

For example, if you want to update your repository every day beginning at 2:30 AM:

```
30 2 * * * rsync -avz rsync://centos.ar.host-engine.com/6.6/os/x86_64/ /var/www/html/repos/centos/6/6/
```

Of course, you can put that line inside a script to do more complex and customized tasks before or after performing the update. Feel free to experiment and tell me about the results.

Summary

You should never underestimate the importance of a local or network repository given the many benefits it brings as I explained in this article. If you can afford the disk space, this is definitely the way to go.

Chapter 32: Network performance, security, and troubleshooting

A sound analysis of a computer network begins by understanding what are the available tools to perform the task, how to pick the right one(s) for each step of the way, and last but not least, where to begin.

In this article we will review some well-known tools to examine the performance and increase the security of a network, and what to do when things aren't going as expected. Please note that this list does not present to be comprehensive, so feel free to comment on this post using the form at the bottom if you would like to add another useful utility that we could be missing.

What services are running and why?

One of the first things that a system administrator needs to know about each system is what services are running and why. With that information in hand, it is a wise decision to disable all those that are not strictly necessary and shun hosting too many servers in the same physical machine. For example, you need to disable your FTP server if your network does not require one (there are more secure methods to share files over a network, by the way). In addition, you should avoid having a web server and a database server in the same system. If one component becomes compromised, the rest run the risk of getting compromised as well.

Investigating socket connections with ss

ss is used to dump socket statistics and shows information similar to netstat, though it can display more TCP and state informations than other tools. In addition, it is listed in man netstat as replacement for netstat, which is obsolete.

However, in this article we will focus on the information related to network security only.

Example 1: Showing ALL TCP ports (sockets) that are open on our server

All services running on their default ports (i.e. http on 80, mysql on 3306) are indicated by their respective names. Others (obscured here for privacy reasons) are shown in their numeric form.

```
ss -t -a
```

```
[gacanepa@dodev01 ~]$ ss -t -a
State       Recv-Q Send-Q                Local Address:Port                 Peer Address:Port
LISTEN      0      100                       127.0.0.1:smtp                        *:*
LISTEN      0      50                              *:mysql                       *:*
LISTEN      0      128                             *:[redacted]                  *:*
ESTAB       0      64                      [redacted]                    [redacted]:49299
LISTEN      0      128                          :::https                      :::*
LISTEN      0      128                          :::http                       :::*
LISTEN      0      128                          :::[redacted]                 :::*
LISTEN      0      32                           :::ftp                        :::*
[gacanepa@dodev01 ~]$
```

The first column shows the TCP state, while the second and third column display the amount of data that is currently queued for reception and transmission. The fourth and fifth columns show the source and destination sockets of each connection. On a side note, you may want to check RFC 793 to refresh your memory about possible TCP states because you also need to check on the number and the state of open TCP connections in order to become aware of (D)DoS attacks.

Example 2: Displaying ALL active TCP connections with their timers

```
ss -t -o
```

```
[gacanepa@dodev01 ~]$ ss -t -o
State       Recv-Q Send-Q                                  Local Address:Port                                  Peer Address:Port
ESTAB       0      0                                                                                                    :49299
timer:(keepalive,36min,0)
ESTAB       0      64                                                                                                   :50073
timer:(on,735ms,0)
[gacanepa@dodev01 ~]$
```

In the output above, you can see that there are 2 established SSH connections. If you notice the value of the second field of timer:, you will notice a value of 36 minutes in the first connection. That is the amount of time until the next keepalive probe will be sent. Since it's a connection that is being kept alive, you can safely assume that is an inactive connection and thus can kill the process after finding out its PID.

```
[root@dodev01 ~]# ps -ef | grep ssh | grep -v grep
root      2377 15863  0 17:15 ?        00:00:00 sshd: gacanepa [priv]
gacanepa  2379  2377  0 17:15 ?        00:00:00 sshd: gacanepa@pts/0
root      2793 15863  0 19:27 ?        00:00:00 sshd: gacanepa [priv]
gacanepa  2795  2793  0 19:27 ?        00:00:00 sshd: gacanepa@pts/1
root     15863     1  0  2014 ?        00:00:00 /usr/sbin/sshd -D
[root@dodev01 ~]# kill 2377
[root@dodev01 ~]# ss -t -o
State       Recv-Q Send-Q                                  Local Address:Port                                  Peer Address:Port
FIN-WAIT-1  0      1
timer:(on,2.623ms,3)
ESTAB       0      64
timer:(on,705ms,0)
[root@dodev01 ~]# ss -t -o
State       Recv-Q Send-Q                                  Local Address:Port                                  Peer Address:Port
FIN-WAIT-1  0      1
timer:(on,5.917ms,5)
ESTAB       0      64
timer:(on,695ms,0)
[root@dodev01 ~]# kill -9 2377
-bash: kill: (2377) - No such process
[root@dodev01 ~]# ps -ef | grep ssh | grep -v grep
root      2793 15863  0 19:27 ?        00:00:00 sshd: gacanepa [priv]
gacanepa  2795  2793  0 19:27 ?        00:00:00 sshd: gacanepa@pts/1
root     15863     1  0  2014 ?        00:00:00 /usr/sbin/sshd -D
[root@dodev01 ~]#
```

Note that the connection state changed to FIN-WAIT-1, which indicates that it's waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.
The process, however, was killed and no longer exists.

As for the second connection, you can see that it's currently being used (as indicated by on).

Example 3: Filtering connections by socket

Suppose you want to filter TCP connections by socket. From the server's point of view, you need to check for connections where the source port is 80

```
ss -tn sport = :80
```

Resulting in

```
[root@dodev01 ~]# ss -tn sport = :80
State       Recv-Q Send-Q                                  Local Address:Port                                  Peer Address:Port
ESTAB       0      0                                                  .101:80                                            .198:50567
ESTAB       0      0                                                  .101:80                                            .198:50568
ESTAB       0      0                                                  .101:80                                            .198:50570
```

Protecting against port scanning with nmap

Port scanning is a common technique used by crackers to identify active hosts and open ports on a network. Once a vulnerability is discovered, it is exploited in order to gain access to the system.

A wise sysadmin needs to check how his or her systems are seen by outsiders, and make sure nothing is left to chance by auditing them frequently. That is called "defensive port scanning".

Example 4: Displaying information about open ports

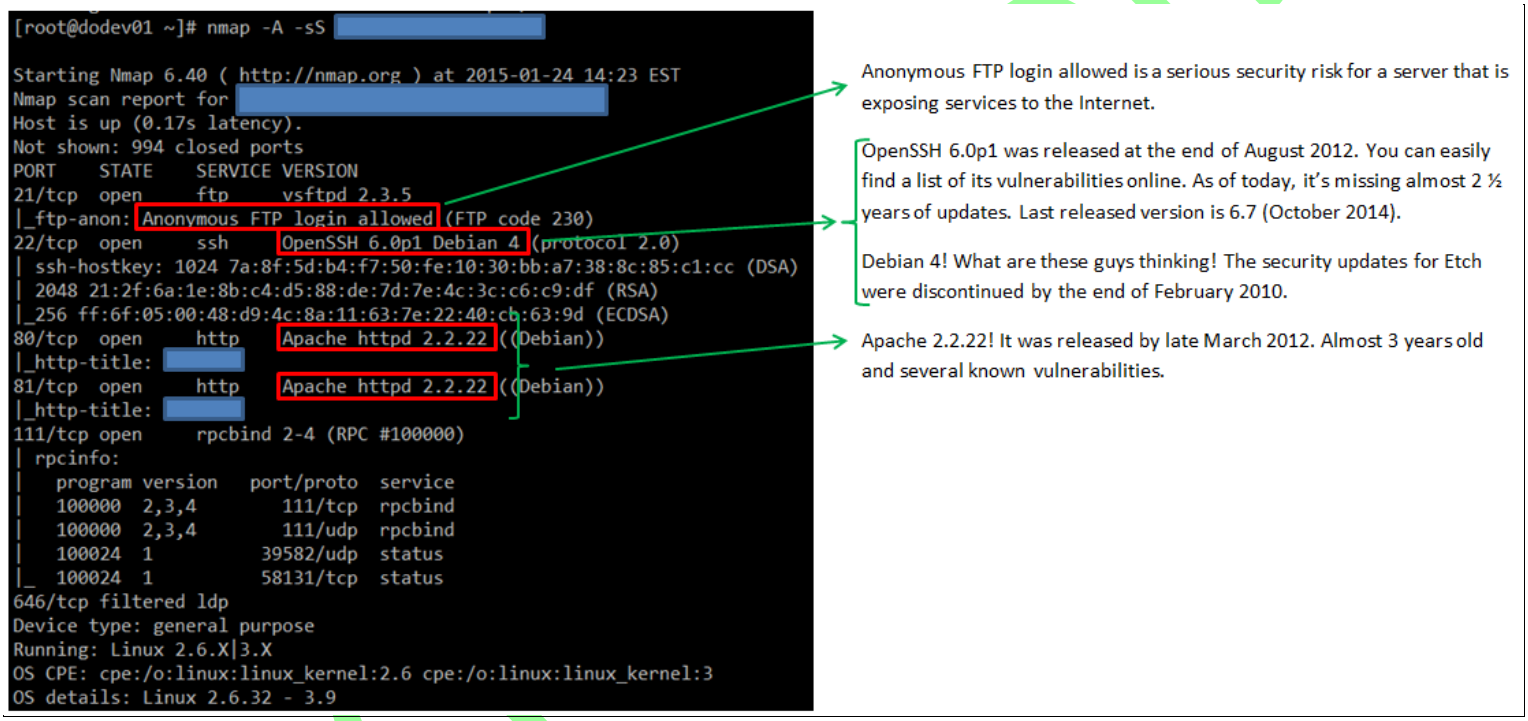
You can use the following command to scan which ports are open on your system or in a remote host:

```
nmap -A -sS [IP address or hostname]
```

The above command will scan the host for OS and version detection, port information, and traceroute (-A). Finally, -sS sends a TCP SYN scan, preventing nmap to complete the 3-way TCP handshake and thus typically leaving no logs on the target machine.

Before proceeding with the next example, please keep in mind that port scanning is not an illegal activity. What IS illegal is using the results for a malicious purpose.

For example, the output of the above command run against the main server of a local university returns the following (only part of the result is shown for sake of brevity):



As you can see, we discovered several anomalies that we should do well to report to the system administrators at this local university.

This specific port scan operation provides all the information that can also be obtained by other commands, such as:

Example 5: Displaying information about a specific port in a local or remote system

```
nmap -p [port] [hostname or address]
```

Example 6: Showing traceroute to, and finding out version of services and OS type, hostname

```
nmap -A [hostname or address]
```

Example 7: Scanning several ports or hosts simultaneously

You can also scan several ports (range) or subnets, as follows:

```
nmap -p 21,22,80 192.168.0.0/24  # Scan ports 21, 22, and 80 on all hosts in that network segment.
```

You can check the man page for further details on how to perform other types of port scanning. Nmap is indeed a very powerful and versatile network mapper utility, and you should be very well acquainted with it in order to defend the systems you're responsible for against attacks originated after a malicious port scan by outsiders.

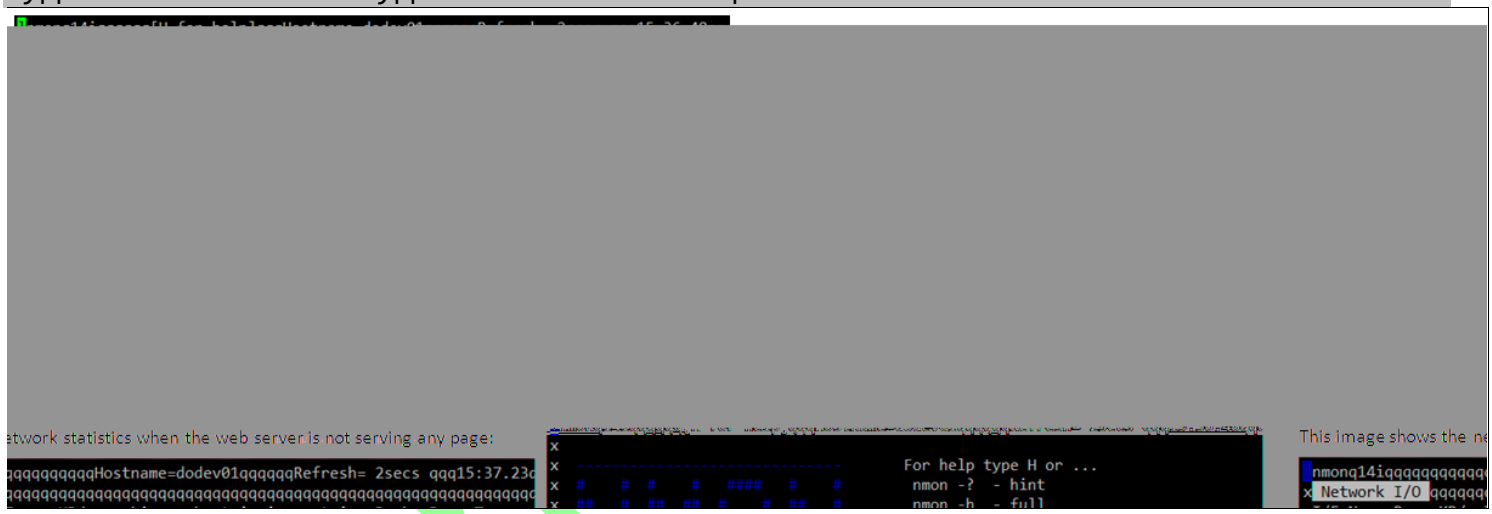
Reporting usage and performance on your network

Although there are several available tools to analyze and troubleshoot network performance, two of them are very easy to learn and user friendly. To install both of them on CentOS, you will need to enable the EPEL repository first.

1) nmon is a system tuner and benchmark tool. As such, it can display the CPU, memory, network, disks, file systems, NFS, top processes, and resources (Linux version & processors). Of course, we're mainly interested in the network performance feature.

To install nmon, run the following command on your chosen distribution:

```
yum update && yum install nmon # CentOS
aptitude update && aptitude install nmon # Ubuntu
zypper refresh && and zypper install nmon # openSUSE
```



Make it a habit to look at the network traffic in real time to ensure that your system is capable of supporting normal loads and to watch out for unnecessary traffic or suspicious activity.

2) **vnstat** is a console-based network traffic monitor that keeps a log of hourly (daily or monthly as well) network traffic for the selected interface(s).

```
yum update && yum install vnstat # CentOS
aptitude update && aptitude install vnstat # Ubuntu
zypper refresh && and zypper install vnstat # openSUSE
```

After installing the package, you need to enable the monitoring daemon as follows:

```
service vnstat start # SysV-based systems (Ubuntu)
systemctl start vnstat # systemd-based systems (CentOS / openSUSE)
```

Once you have installed and enabled vnstat, you can initialize the database to record traffic for eth0 (or other NIC) as follows:

```
vnstat -u -i eth0
```

As I have just installed vnstat in the machine that I'm using to write this article, I still haven't gathered enough data to display usage statistics:

```
[root@dodev01 ~]# vnstat
 eth0: Not enough data available yet.
[root@dodev01 ~]#
```

The vnstatd daemon will continue running in the background and collecting traffic data. Until it collects enough data to produce output, you can refer to the project's web site to see what the traffic analysis looks like.

Transferring files securely over the network

If you need to ensure security while transferring or receiving files over a network, and specially if you need to perform that operation over the Internet, you will want to resort to 2 secure methods for file transfers (don't even think about doing it over plain FTP!).

Example 8: Transferring files with scp (secure copy)

Use the -P flag if SSH on the remote hosts is listening on a port other than the default 22. The -p switch will preserve the permissions of local_file after the transfer, which will be made with the credentials of remote_user on remote_hosts. You will need to make sure that /absolute/path/to/remote/directory is writeable by this user.

```
scp -P XXXX -p local_file remote_user@remote_host:/absolute/path/to/remote/directory
```

Example 9: Receiving files with scp (secure copy)

You can also download files with scp from a remote host:

```
scp remote_user@remote_host:myFile.txt /absolute/path/to/local/directory
```

Or even between two remote hosts (in this case, copy the file myFile.txt from remote_host1 to remote_host2):

```
scp remote_user1@remote_host1:/absolute/path/to/remote/directory1/myFile.txt
remote_user1@remote_host2:/absolute/path/to/remote/directory2/
```

Don't forget to use the -P switch if SSH is listening on a port other than the default 22.

You can read more about SCP here.

Example 10: Sending and receiving files with SFTP

Unlike SCP, SFTP does not require previously knowing the location of the file that we want to download or send.

This is the basic syntax to connect to a remote host using SFTP:

```
sftp -oPort=XXXX username@host
```

where XXXX represents the port where SSH is listening on host, which can be either a hostname or its corresponding IP address. You can disregard the -oPort flag if SSH is listening on its default port (22).

Once the connection is successful, you can issue the following commands to send or receive files:

```
get -Pr [remote file or directory] # Receive files
put -r [local file or directory] # Send files
```

In both cases, the -r switch is used to recursively receive or send files, respectively. In the first case, the -P option will also preserve the original file permissions.

To close the connection, simply type "exit" or "bye". You can read more about sftp here.

Summary

You may want to complement what we have covered in this article with what we've already learned in other chapters. If you know your systems well, you will be able to easily detect malicious or suspicious activity when the numbers show unusual activity without an apparent reason. You will also be able to plan ahead for network resources if you're expecting a sudden increase in their use.